

LNCS 5484

Mario Giacobini et al. (Eds.)

Applications of Evolutionary Computing

EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT
EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION
EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG
Tübingen, Germany, April 2009, Proceedings



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Mario Giacobini et al. (Eds.)

Applications of Evolutionary Computing

EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT
EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION
EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG
Tübingen, Germany, April 15-17, 2009
Proceedings



Springer

Volume Editors
see next page

Cover illustration: "You Pretty Little Flocker" by Alice Eldridge (www.ecila.org
and www.infotech.monash.edu.au/research/groups/cema/flocker/flocker.html)

Library of Congress Control Number: Applied for

CR Subject Classification (1998): F.1, D.1, B, C.2, J.3, I.4, J.5

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743

ISBN-10 3-642-01128-4 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-01128-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12651801 06/3180 5 4 3 2 1 0

Volume Editors

Mario Giacobini
University of Torino, Italy
mario.giacobini@unito.it

Anthony Brabazon
University College Dublin, Ireland
anthony.brabazon@ucd.ie

Stefano Cagnoni
University of Parma, Italy
cagnoni@ce.unipr.it

Gianni A. Di Caro
IDSIA, Lugano, Switzerland
gianni@idsia.ch

Anikó Ekárt
Aston University, Birmingham, UK
ekarta@aston.ac.uk

Anna I Esparcia-Alcázar
Instituto Tecnológico de Informática
Valencia, Spain
anna@iti.upv.es

Muddassar Farooq
National University of Computer
and Emerging Sciences, Pakistan
muddassar.farooq@nu.edu.pk

Andreas Fink
Helmut-Schmidt-University
Hamburg, Germany
andreas.fink@hsu-hamburg.de

Penousal Machado
University of Coimbra, Portugal
machado@dei.uc.pt

Jon McCormack
Monash University, Clayton, Australia
Jon.McCormack@infotech.monash.edu.au

Michael O'Neill
University College Dublin, Ireland
m.oneill@ucd.ie

Ferrante Neri
University of Jyväskylä, Finland
neferran@cc.jyu.fi

Mike Preuss
TU Dortmund University, Germany
mike.preuss@tu-dortmund.de

Franz Rothlauf
Johannes Gutenberg University
Mainz, Germany
rothlauf@uni-mainz.de

Ernesto Tarantino
ICAR - CNR, Naples, Italy
ernesto.tarantino@na.icar.cnr.it

Shengxiang Yang
University of Leicester, UK
s.yang@mcs.le.ac.uk

Preface

The year 2009 celebrates the bicentenary of Darwin's birth and the 150th anniversary of the publication of his seminal work, *On the Origin of Species*. If this makes 2009 a special year for the research community working in biology and evolution, the field of evolutionary computation (EC) also shares the same excitement. EC techniques are efficient, nature-inspired planning and optimization methods based on the principles of natural evolution and genetics. Due to their efficiency and simple underlying principles, these methods can be used in the context of problem solving, optimization, and machine learning. A large and ever-increasing number of researchers and professionals make use of EC techniques in various application domains.

This volume presents a careful selection of relevant EC applications combined with a thorough examination of the techniques used in EC. The papers in the volume illustrate the current state of the art in the application of EC and can help and inspire researchers and professionals to develop efficient EC methods for design and problem solving.

All the papers in this book were presented during the 2009 edition of EvoWorkshops, which was held at the Eberhard Karls Universität Tübingen, Germany, during April 15–17, 2009. EvoWorkshops are composed of a range of workshops on application-oriented aspects of EC that, since 1998, has provided a unique opportunity for EC researchers to meet and discuss application aspects of EC and has been an important link between EC research and its application in a variety of domains. During these past ten years, several workshops have been proposed, some of them have disappeared along the way, while others have matured to become conferences of their own, such as EuroGP in 2000, EvoCOP in 2004, and EvoBIO two years ago.

EvoWorkshops are part of EVO*, Europe's premier co-located events in the field of EC. EVO* includes, in addition to EvoWorkshops, EuroGP, the main European event dedicated to genetic programming; EvoCOP, the main European conference on EC in combinatorial optimization; and EvoBIO, the main European conference on EC and related techniques in bioinformatics and computational biology. The proceedings for all of these events, EuroGP 2009, EvoCOP 2009, and EvoBIO 2009, are also available in the LNCS series (volumes 5481, 5482, and 5483).

The central aim of the EVO* events is to provide researchers, as well as people from industry, students, and interested newcomers, with an opportunity to present new results, discuss current developments and applications, or just become acquainted with the world of EC. Moreover, it encourages and reinforces possible synergies and interactions between members of all scientific communities that may benefit from EC techniques.

EvoWorkshops 2009 consisted of the following individual workshops:

- *EvoCOMNET*, the Sixth European Workshop on the Application of Nature-Inspired Techniques for Telecommunication Networks and Other Parallel and Distributed Systems,
- *EvoENVIRONMENT*, the First European Workshop on Nature-Inspired Methods for Environmental Issues,
- *EvoFIN*, the Third European Workshop on Evolutionary Computation in Finance and Economics,
- *EvoGAMES*, the First European Workshop on Bio-inspired Algorithms in Games,
- *EvoHOT*, the Fifth European Workshop on Bio-Inspired Heuristics for Design Automation,
- *EvoIASP*, the Eleventh European Workshop on Evolutionary Computation in Image Analysis and Signal Processing,
- *EvoINTERACTION*, the Third European Workshop on Interactive Evolution and Humanized Computational Intelligence,
- *EvoMUSART*, the Seventh European Workshop on Evolutionary and Biologically Inspired Music, Sound, Art and Design,
- *EvoNUM*, the Second European Workshop on Bio-inspired Algorithms for Continuous Parameter Optimization,
- *EvoSTOC*, the Sixth European Workshop on Evolutionary Algorithms in Stochastic and Dynamic Environments, and
- *EvoTRANSLOG*, the Third European Workshop on Evolutionary Computation in Transportation and Logistics.

EvoCOMNET addresses the application of EC techniques to the definition, analysis, and development of novel parallel and distributed algorithms, and to the solution of problems of practical and theoretical interest in all domains related to network systems. To address these challenges, this workshop promotes the study and the application of strategies inspired by the observation of biological and evolutionary processes, which usually show the highly desirable characteristics of being distributed, adaptive, scalable, and robust.

EvoENVIRONMENT is devoted to the use of nature-inspired methods for environmental issues. It deals with many diverse topics such as waste management, sewage treatment, control of greenhouse gas emissions, biodegradation of materials, efficient energy use, or use of renewable energies, to name but a few.

EvoFIN is the only European event specifically dedicated to the applications of EC, and related natural computing methodologies, to finance and economics. Financial environments are typically hard, being dynamic, high-dimensional, noisy, and co-evolutionary. These environments serve as an interesting test bed for novel evolutionary methodologies.

EvoGAMES aims to focus the scientific developments onto computational intelligence techniques that may be of practical value for utilization in existing

or future games. Recently, games, and especially video games, have become an important commercial factor within the software industry, providing an excellent test bed for application of a wide range of computational intelligence methods.

EvoHOT focuses on innovative heuristics, game theory and bio-inspired techniques applied to the electronic design automation. It shows the latest developments, the reports of industrial experiences, the successful attempts to evolve rather than design new solutions, and the hybridizations of traditional methodologies.”

EvoIASP, the longest-running of all EvoWorkshops, which celebrates its eleventh edition this year, has been the first international event solely dedicated to the applications of EC to image analysis and signal processing in complex domains of high industrial and social relevance.

EvoINTERACTION deals with various aspects of interactive evolution, and more broadly of computational intelligence in interaction with human intelligence, including methodology, theoretical issues, and new applications. Interaction with humans raises several problems, mainly linked to what has been called the user bottleneck, i.e., the human fatigue.

EvoMUSART addresses all practitioners interested in the use of EC techniques for the development of creative systems. There is a growing interest in the application of these techniques in fields such as art, music, architecture, and design. The goal of this workshop is to bring together researchers that use EC in this context, providing an opportunity to promote, present, and discuss the latest work in the area, fostering its further developments and collaboration among researchers.

EvoNUM aims at applications of bio-inspired algorithms, and cross-fertilization between these and more classical numerical optimization algorithms, to continuous optimization problems in engineering. It deals with engineering applications where continuous parameters or functions have to be optimized, in fields such as control, chemistry, agriculture, electricity, building and construction, energy, aerospace engineering, and design optimization.

EvoSTOC addresses the application of EC in stochastic and dynamic environments. This includes optimization problems with changing, noisy, and/or approximated fitness functions and optimization problems that require robust solutions. These topics recently gained increasing attention in the EC community and EvoSTOC was the first workshop that provided a platform to present and discuss the latest research in this field.

EvoTRANSLOG deals with all aspects of the use of EC, local search, and other nature-inspired optimization and design techniques for the transportation and logistics domain. The impact of these problems on the modern economy and society has been growing steadily over the last few decades, and the workshop aims at design and optimization techniques such as evolutionary computing approaches allowing the use of computer systems for systematic design, optimization, and improvement of systems in the transportation and logistics domain.

EvoWorkshops adapt annually to the needs and preferences of researchers, and some workshops were not run this year. EvoTHEORY, the European

Workshop on Theoretical Aspects in Artificial Evolution, decided not to run in 2008 and will run again next year. While it did not take place during EVO* 2008, EvoINTERACTION was held this year, and two new workshops were also proposed this year, EvoENVIRONMENT and EvoGAMES.

The number of submissions to EvoWorkshops 2009 was once again very high, cumulating 143 entries (compared to 160 in 2007 and 133 in 2008). The following table shows relevant statistics for EvoWorkshops 2009 (both short and long papers are considered in the acceptance statistics), where also the statistics for the 2008 edition are reported, except for EvoINTERACTION whose last edition was in 2007:

Workshop	2009			Previous Edition		
	Submit	Accept	Rate	Submit	Accept	Rate
EvoCOMNET	21	15	71.4%	10	6	60%
EvoENVIRONMENT	5	4	80%	-	-	-
EvoFIN	14	8	57.1%	15	8	53.3%
EvoGAMES	15	10	66.6%	-	-	-
EvoHOT	5	4	80%	15	9	60%
EvoIASP	14	7	50%	26	16	61.5%
EvoINTERACTION	5	4	80%	7	4	57.1%
EvoMUSART	26	17	65.3%	31	17	54.8%
EvoNUM	16	9	56.2%	14	8	57.1%
EvoSTOC	11	7	63.6%	8	4	50%
EvoTHEORY	-	-	-	3	2	66.6
EvoTRANSLOG	11	6	54.5%	11	5	45.4%
Total	143	91	63.6%	133	75	56.4%

The events accepted ten-page full papers and six-page short papers. The two classes of papers were either presented orally over the three conference days, or presented and discussed during a special poster session. The significant number of submissions for EvoWorkshops 2009 shows the liveliness of the scientific activities in the corresponding fields.

Many people helped make EvoWorkshops a success. We would like to thank the following institutions:

- The Eberhard Karls Universität Tübingen, Germany
- The German Research Foundation (DFG) for financial support
- The Centre for Emergent Computing at Edinburgh Napier University, Scotland, for administrative help and event coordination
- The Institute for High Performance Computing and Networking of the Italian National Research Council

We want to especially acknowledge the invited speakers that gave two very interesting and inspirational talks during the conference days: Peter Schuster, President of the Austrian Academy of Sciences, and Stuart R. Hameroff, Professor Emeritus, Departments of Anesthesiology and Psychology and Director, Center for Consciousness Studies, University of Arizona, Tucson, Arizona.

We are also very grateful to all the people who provided local support, and in particular to Andreas Zell, Chair of Computer Architecture at the Wilhelm-Schickard Institute for Computer Science at the University of Tübingen, Peter Weit, Vice Director of the Seminar for Rhetorics at the New Philology Department at the University of Tübingen, and the Tourist Information Center of Tübingen, especially Marco Schubert.

Even with an excellent support and location, an event like EVO* would not have been feasible without authors submitting their work, members of the Program Committees dedicating energy in reviewing those papers, and an audience. All these people deserve our gratitude.

Finally, we are grateful to all those involved in the preparation of the event, especially Jennifer Willies for her unfaltering dedication to the coordination of the event over the years. Without her support, running this type of conference, with a large number of different organizers and different opinions, would be unmanageable. Further thanks to the Local Chair Marc Ebner for making the organization of such an event possible and successful. Last but surely not least, we want to especially acknowledge Ivano De Falco for his hard work as Publicity Chair of the event, and Marc Schoenauer for his continuous help in setting up and maintaining the MyReview conference management software.

April 2009

Mario Giacobini	Penousal Machado
Anthony Brabazon	Jon McCormack
Stefano Cagnoni	Ferrante Neri
Gianni A. Di Caro	Michael O'Neill
Anikó Ekárt	Mike Preuss
Anna I. Esparcia-Alcázar	Franz Rothlauf
Muddassar Farooq	Ernesto Tarantino
Andreas Fink	Shengxiang Yang

Organization

EvoWorkshops 2009, together with the conferences EuroGP 2009, EvoCOP 2009, and EvoBIO 2009, was part of EVO* 2009, Europe's premier co-located events in the field of evolutionary computing.

Organizing Committee

EvoWorkshops Chair	Mario Giacobini, University of Torino, Italy
Local Chair	Marc Ebner, Eberhard Karls Universität Tübingen, Germany
Publicity Chair	Ivanoe De Falco, ICAR, CNR, Italy
EvoCOMNET Co-chairs	Gianni A. Di Caro, IDSIA, Switzerland Muddassar Farooq, National University of Computer and Emerging Sciences, Pakistan Ernesto Tarantino, ICAR, CNR, Italy
EvoENVIRONMENT	
Co-chairs	Marc Ebner, Eberhard Karls Universität Tübingen, Germany Neil Urquhart, Edinburgh Napier University, UK
EvoFIN Co-chairs	Anthony Brabazon, University College Dublin, Ireland Michael O'Neill, University College Dublin, Ireland
EvoGAMES Co-chairs	Mike Preuss, TU Dortmund University, Germany Anna Isabel Esparcia-Alcazar, Universitat Politècnica de València, Spain
EvoHOT Co-chairs	Rolf Drechsler, Universität Bremen, Germany Giovanni Squillero, Politecnico di Torino, Italy
EvoIASP Chair	Stefano Cagnoni, University of Parma, Italy
EvoINTERACTION	
Co-chairs	Evelyne Lutton, INRIA, France Hideyuki Takagi, Kyushu University, Japan
EvoMUSART Co-chairs	Penousal Machado, University of Coimbra, Portugal Jon McCormack, Monash University, Australia

EvoNUM Co-chairs	Anna Isabel Esparcia-Alcazar, Universitat Politécnica de València, Spain
	Anikó Ekárt, Aston University, UK
EvoSTOC Co-chairs	Shengxiang Yang, University of Leicester, UK
	Ferrante Neri, University of Jyväskylä, Finalnd
EvoTRANSLOG Co-chairs	Andreas Fink, Helmut Schmidt University Hamburg, Germany
	Franz Rothlauf, Johannes Gutenberg University Mainz, Germany

Program Committees

EvoCOMNET Program Committee

Uwe Aickelin	University of Nottingham, UK
Özgür B. Akan	Middle East Technical University, Turkey
Jarmo Alander	Helsinki University of Technology, Finland
Mehmet E. Aydin	University of Bedfordshire, UK
Arindam K. Das	University of Washington, USA
Falko Dressler	University of Erlangen, Germany
Frederick Ducatelle	IDSIA, Switzerland
Jin-Kao Hao	University of Angers, France
Malcolm I. Heywood	Dalhousie University, Canada
Kenji Leibnitz	Osaka University, Japan
Manuel Lozano Marquez	University of Granada, Spain
Domenico Maistro	University of Modena-Reggio Emilia, Italy
Vittorio Maniezzo	University of Bologna, Italy
Roberto Montemanni	IDSIA, Switzerland
Umberto Scafuri	ICAR-CNR, Italy
Chien-Chung Shen	University of Delaware, USA
Kwang M. Sim	Hong Kong Baptist University, Hong Kong
Luigi Troiano	University of Sannio, Italy
Lidia Yamamoto	University of Basel, Switzerland

EvoENVIRONMENT Program Committee

Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Roland Benz	Universität Würzburg, Germany
Paul Brunner	Technische Universität Wien, Austria
Stefano Cagnoni	University of Parma, Italy
Wolfgang Calmano	Hamburg University of Technology, Germany
Pierre Collet	Louis Pasteur University of Strasbourg, France
Bart Craenen	Napier University, UK
Kevin Cullinane	Napier University, UK
Sharon Cullinane	Heriot-Watt University, UK

Peter Dittrich	Friedrich Schiller University Jena, Germany
Marc Ebner	Eberhard Karls Universität Tübingen, Germany
Anikó Ekárt	Aston University, UK
Ali Elkamel	University of Waterloo, Canada
Anna I Esparcia-Alcázar	Instituto Tecnológico de Informática, Spain
Stephen Evans	Cranfield University, UK
James A Foster	University of Idaho, USA
Satoshi Ishii	The University of Tokyo, Japan
Christian Jacob	University of Calgary, Canada
Rhyd Lewis	Cardiff Business School, UK
Rongxin Li	CSIRO ICT Centre, Australia
William Magette	University College Dublin, Ireland
R.I. (Bob) McKay	Seoul National University, Korea
Julian F Miller	University of York, UK
Michael O'Neill	University College Dublin, Ireland
Sharon Perez-Suarez	Environmental Protection Agency, USA
Conor Ryan	University of Limerick, Ireland
Tom Rye	Napier University, UK
Carlo Santulli	University of Rome “La Sapienza”, Italy
Marc Schoenauer	INRIA, France
Terence Soule	University of Idaho, USA
Neil Urquhart	Napier University, UK
Jano van Hemert	University of Edinburgh, UK
Tina Yu	Memorial University of Newfoundland, Canada
Mengjie Zhang	University of Wellington, New Zealand

EvoFIN Program Committee

Eva Alfaro-Cidc	Instituto Tecnológico de Informática Valencia, Spain
Anthony Brabazon	University College Dublin, Ireland
Ian Dempsey	Pipeline Trading, USA
Rafał Dreżewski	AGH University of Science and Technology, Poland
Kai Fan	University College Dublin, Ireland
Philip Hamill	University of Ulster, UK
Ronald Hochreiter	University of Vienna, Austria
Youwei Li	Queen’s University Belfast, UK
Dietmar Maringer	University of Basel, Switzerland
Michael O'Neill	University College Dublin, Ireland
Conal O’Sullivan	University College Dublin, Ireland

Robert Schafer	AGH University of Science and Technology, Poland
Chris Stephens	Universidad Nacional Autonoma de Mexico, Mexico
Andrea Tettamanzi	Universita Degli Studi di Milano, Italy

EvoGAMES Program Committee

Lourdes Araujo	UNED, Spain
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Luigi Barone	University of Western Australia, Australia
Nicola Beume	TU Dortmund University, Germany
Simon Colton	Imperial College London, UK
Ernesto Costa	Universidade de Coimbra, Portugal
Carlos Cotta	Universidad de Málaga, Spain
Marc Ebner	Universität Tübingen, Germany
Anikó Ekárt	Aston University, UK
Antonio J. Fernández Leiva	Universidad de Málaga, Spain
Francisco Fernández	Universidad de Extremadura, Spain
Mario Giacobini	Università degli Studi di Torino, Italy
David Hart	Fall Line Studio, USA
Philip Hingston	Edith Cowan University, Australia
Krzysztof Krawiec	Poznan University of Technology, Poland
Oliver Kramer	TU Dortmund University, Germany
Bill Langdon	University of Essex, UK
Simon Lucas	University of Essex, UK
Penousal Machado	Universidade de Coimbra, Portugal
Juan Julián Merelo	Universidad de Granada, Spain
Steffen Priesterjahn	University of Paderborn, Germany
Moshe Sipper	Ben-Gurion University, Israel
Terry Soule	University of Idaho, USA
Julian Togelius	IDSIA, Switzerland
Georgios N Yannakakis	IT University of Copenhagen, Denmark

EvoHOT Program Committee

Varun Aggarwal	MIT, Cambridge, USA
Michelangelo Grosso	Politecnico di Torino, Italy
Doina Logofatu	University of Applied Sciences, Munich, Germany
Gustavo Olague	CICESE Research Center, San Diego, USA
Mihai Oltean	Babeş-Bolyai University, Cluj-Napoca, Romania
Gregor Papa	Jozef Stefan Institute, Ljubljana, Slovenia
Wilson Javier Pérez Holguín	Universidad de Valle, Cali, Colombia

Danilo Ravotto	Politecnico di Torino, Italy
Ernesto Sanchez	Politecnico di Torino, Italy
Massimiliano Schillaci	Politecnico di Torino, Italy

EvoIASP Program Committee

Lucia Ballerini	University of Edinburgh, UK
Bir Bhanu	University of California at Riverside, USA
Leonardo Bocchi	University of Florence, Italy
Ela Claridge	University of Birmingham, UK
Oscar Cordon	European Center for Soft Computing, Spain
Ivanoe De Falco	ICAR CNR, Italy
Antonio Della Cioppa	University of Salerno, Italy
Laura Dipietro	Massachusetts Institute of Technology, USA
Marc Ebner	University of Tübingen, Germany
Špela Ivetković	University of Dundee, UK
Mario Köppen	Kyuhu Institute of Technology, Japan
Krzysztof Krawiec	Poznan University of Technology, Poland
Evelyne Lutton	INRIA, France
Luca Mussi	University of Parma, Italy
Ferrante Neri	University of Jyväskylä, Finland
Gustavo Olague	CICESE, Mexico
Riccardo Poli	University of Essex, UK
Stephen Smith	University of York, UK
Giovanni Squillero	Politecnico di Torino, Italy
Kiyoshi Tanaka	Shinshu University, Japan
Ankur M. Teredesai	University of Washington Tacoma, USA
Andy Tyrrell	University of York, UK
Leonardo Vanneschi	University of Milano-Bicocca, Italy
Mengjie Zhang	Victoria University of Wellington, New Zealand

EvoINTERACTION Program Committee

Breanna Bailey	University-Kingsville, USA
Eric Bonabeau	Icosystem, USA
Larry Bull	UWE Bristol, UK
Praminda Caleb-Solly	University of the West of England, UK
Pierre Collet	University of Strasbourg, France
Ian Graham	Loughborough University, UK
Pr Fang-Cheng Hsu	Aletheia University, China
Christian Jacob	University of Calgary, Canada
Yaochu Jin	Honda Research Institute Europe, Germany
Daisuke Katagami	Tokyo Institute of Technology, Japan
Penousal Machado	University of Coimbra, Spain
Yoichiro Maeda	University of Fukui, Japan
Nicolas Monmarche	Université de Tours, France

XVIII Organization

Hiroaki Nishino	Oita University, Japan
Ian C. Parmee	UWE Bristol, UK
Yago Saez	Universidad CARLOS III de Madrid, Spain
Marc Schoenauer	INRIA, France
Marc Shackelford	UWE Bristol, UK
Leuo-hong Wang	Aletheia University, China

EvoMUSART Program Committee

Mauro Annunziato	Plancton Art Studio, Italy
Peter Bentley	University College London, UK
Eleonora Bilotta	University of Calabria, Italy
Jon Bird	University of Sussex, UK
Tim Blackwell	Goldsmiths, University of London, UK
Oliver Bown	Monash University, Australia
Paul Brown	University of Sussex, UK
Stefano Cagnoni	University of Parma, Italy
Amílcar Cardoso	University of Coimbra, Portugal
John Collomosse	University of Bath, UK
Simon Colton	Imperial College, UK
Palle Dahlstedt	Göteborg University, Sweden
Hans Dehlinger	Independent Artist, Germany
Steve DiPaola	Simon Fraser University, Canada
Alan Dorin	Monash University, Australia
Erwin Driessens	Independent Artist, The Netherlands
Carla Farsi	University of Colorado, USA
Philip Galanter	Texas A&M College of Architecture, USA
Pablo Gervás	Universidad Complutense de Madrid, Spain
Andrew Gildfind	Google, Inc., Australia
Gary Greenfield	University of Richmond, USA
Carlos Grilo	Instituto Politécnico de Leiria, Portugal
David Hart	Independent Artist, USA
Andrew Horner	University of Science & Technology, Hong Kong
Christian Jacob	University of Calgary, Canada
Colin Johnson	University of Kent, UK
William Latham	Goldsmiths, University of London, UK
Matthew Lewis	Ohio State University, USA
Alain Lioret	Paris 8 University, France
Bill Manaris	College of Charleston, USA
Ruli Manurung	University of Indonesia, Indonesia
Jonatas Manzolli	UNICAMP, Brazil
James McDermott	University of Limerick, Ireland
Nicolas Monmarché	University of Tours, France
Gary Nelson	Oberlin College, USA
Luigi Pagliarini	Pescara Electronic Artists Meeting, Italy and University of Southern Denmark, Denmark

Alejandro Pazos Somnuk	University of A Coruña, Spain
Phon-Amnuaisuk	Multimedia University, Malaysia
Rafael Ramirez	Pompeu Fabra University, Spain
Juan Romero	University of A Coruña, Spain
Brian Ross	Brock University, Canada
Artemis Sanchez Moroni	Renato Archer Research Center, Brazil
Antonino Santos	University of A Coruna, Spain
Kenneth O. Stanley	University of Central Florida, USA
Jorge Tavares	MIT USA
Stephen Todd	IBM, UK
Paulo Urbano	Universidade de Lisboa , Portugal
Anna Ursyn	University of Northern Colorado, USA
Maria Verstappen	Independent Artist, The Netherlands
Rodney Waschka II	North Carolina State University, USA
Gerhard Widmer	Johannes Kepler University Linz, Austria

EvoNUM Program Committee

Eva Alfaro	Instituto Tecnológico de Informática, Spain
Anne Auger	INRIA, France
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Hans-Georg Beyer	FH Vorarlberg, Austria
Xavier Blasco	Universitat Politècnica de València, Spain
Ying-Ping Chen	National Chiao Tung University, Taiwan
Carlos Cotta	Universidad de Málaga, Spain
Kalyanmoy Deb	Helsinki School of Economics, Finland
Marc Ebner	Universität Tübingen, Germany
Francisco Fernández	Universidad de Extremadura, Spain
Nikolaus Hansen	INRIA, France
William B Langdon	University of Essex, UK
JJ Merelo	Universidad de Granada, Spain
Boris Naujoks	TU Dortmund University, Germany
Gabriela Ochoa	University of Nottingham, UK
Una-May O'Reilly	MIT, USA
Mike Preuss	TU Dortmund University, Germany
Marc Schoenauer	INRIA, France
Günter Rudolph	TU Dortmund University, Germany
Hans-Paul Schwefel	TU Dortmund University, Germany
PN Suganthan	Nanyang Technological University, Singapore
Ke Tang	University of Science and Technology of China, China
Darrell Whitley	Colorado State University, USA

EvoSTOC Program Committee

Dirk Arnold	Dalhousie University, Canada
Thomas Bartz-Beielstein	Cologne University of Applied Sciences, Germany
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Tim Blackwell	Goldsmiths College London, UK
Peter Bosman	Centre for Mathematics and Computer Science, The Netherlands
Juergen Branke	University of Karlsruhe, Germany
Andrea Caponio	Technical University of Bari, Italy
Hui Cheng	University of Leicester, UK
Ernesto Costa	University of Coimbra, Portugal
Kalyanmoy Deb	Indian Institute of Technology Kanpur, India
Anna I Esparcia-Alcazar	Instituto Tecnologico de Informatica, Spain
Chi-Keong Goh	Data Storage Institute, Singapore
Yaochu Jin	Honda Research Institute Europe, Germany
Anna Kononova	University of Leeds, UK
Jouni Lampinen	University of Vaasa, Finland
Xiaodong Li	RMIT University, Australia
Meng-Hiot Lim	Nanyang Technological University, Singapore
Timo Mantere	University of Vaasa, Finland
Daniel Merkle	University of Southern Denmark, Denmark
Zbigniew Michalewicz	University of Adelaide, Australia
Kaisa Miettinen	University of Jyväskylä, Finland
Ronald Morrison	Mitretek Systems, Inc., USA
Yew-Soon Ong	Nanyang Technological University, Singapore
William Rand	Northwestern University, USA
Hendrik Richter	University of Leipzig, Germany
Philipp Rohlfshagen	University of Birmingham, UK
Anabela Simoes	University of Coimbra, Portugal
Kay Chen Tan	National University of Singapore, Singapore
Renato Tinós	Universidade de Sao Paulo, Brazil
Ville Tirronen	University of Jyväskylä, Finland
Şima Uyar	Istanbul Technical University, Turkey
Gary Y. Yen	Oklahoma State University, USA
Qingfu Zhang	University of Essex, UK
Aimin Zhou	University of Essex, UK

EvoTRANSLOG Program Committee

Marco Caserta	University of Hamburg, Germany
Karl Doerner	University of Vienna, Austria
Hoong Chuin Lau	Singapore Management University, Singapore
Christian Prins	University of Technology of Troyes, France
Kay Chen Tan	National University of Singapore, Singapore

Theodore Tsekeris	Center of Planning and Economic Research, Athens, Greece
Stefan Voß	University of Hamburg, Germany

Sponsoring Institutions

- Eberhard Karls Universität Tübingen
- German Research Foundation (DFG)
- The Centre for Emergent Computing, Edinburgh Napier University, UK
- The Institute for High Performance Computing and Networking of the Italian National Research Council

Table of Contents

EvoCOMNET Contributions

Web Application Security through Gene Expression Programming	1
<i>Jaroslaw Skaruz and Franciszek Seredyński</i>	
Location Discovery in Wireless Sensor Networks Using a Two-Stage Simulated Annealing	11
<i>Guillermo Molina and Enrique Alba</i>	
Wireless Communications for Distributed Navigation in Robot Swarms	21
<i>Gianni A. Di Caro, Frederick Ducatelle, and Luca M. Gambardella</i>	
An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks	31
<i>Fatma Corut Ergin, Aysegül Yayımlı, and Şima Uyar</i>	
Extremal Optimization as a Viable Means for Mapping in Grids	41
<i>Ivanoe De Falco, Antonio Della Cioppa, Domenico Maisto, Umberto Scafuri, and Ernesto Tarantino</i>	
Swarm Intelligence Inspired Multicast Routing: An Ant Colony Optimization Approach	51
<i>Xiao-Min Hu, Jun Zhang, and Li-Ming Zhang</i>	
A Framework for Evolutionary Peer-to-Peer Overlay Schemes	61
<i>Michele Amoretti</i>	
Multiuser Scheduling in HSDPA with Particle Swarm Optimization	71
<i>Mehmet E. Aydin, Raymond Kwan, Cyril Leung, and Jie Zhang</i>	
Efficient Signal Processing and Anomaly Detection in Wireless Sensor Networks	81
<i>Markus Wälchli and Torsten Braun</i>	
Peer-to-Peer Optimization in Large Unreliable Networks with Branch-and-Bound and Particle Swarms	87
<i>Balázs Báñhelyi, Marco Biazzini, Alberto Montresor, and Márk Jelasity</i>	
Evolving High-Speed, Easy-to-Understand Network Intrusion Detection Rules with Genetic Programming	93
<i>Agustín Orfila, Juan M. Estevez-Tapiador, and Arturo Ribagorda</i>	

Soft Computing Techniques for Internet Backbone Traffic Anomaly Detection	99
<i>Antonia Azzini, Matteo De Felice, Sandro Meloni, and Andrea G.B. Tettamanzi</i>	
Testing Detector Parameterization Using Evolutionary Exploit Generation	105
<i>Hilmi G. Kayacik, A. Nur Zincir-Heywood, Malcolm I. Heywood, and Stefan Burschka</i>	
Ant Routing with Distributed Geographical Localization of Knowledge in Ad-Hoc Networks	111
<i>Michał Kudelski and Andrzej Pacut</i>	
Discrete Particle Swarm Optimization for Multiple Destination Routing Problems	117
<i>Zhi-hui Zhan and Jun Zhang</i>	
EvoENVIRONMENT Contributions	
Combining Back-Propagation and Genetic Algorithms to Train Neural Networks for Ambient Temperature Modeling in Italy	123
<i>Francesco Ceravolo, Matteo De Felice, and Stefano Pizzuti</i>	
Estimating the Concentration of Nitrates in Water Samples Using PSO and VNS Approaches	132
<i>Pablo López-Espí, Sancho Salcedo-Sanz, Á.M. Pérez-Bellido, Emilio G. Ortiz-García, Oscar Alonso-Garrido, and Antonio Portilla-Figueras</i>	
Optimal Irrigation Scheduling with Evolutionary Algorithms	142
<i>Michael de Paly and Andreas Zell</i>	
Adaptive Land-Use Management in Dynamic Ecological System	152
<i>Nanlin Jin, Daniel S. Chapman, and Klaus Hubacek</i>	
EvoFIN Contributions	
Evolutionary Money Management	162
<i>Philip Saks and Dietmar Maringer</i>	
Prediction of Interday Stock Prices Using Developmental and Linear Genetic Programming	172
<i>Garnett Wilson and Wolfgang Banzhaf</i>	
An Introduction to Natural Computing in Finance	182
<i>Jing Dang, Anthony Brabazon, David Edelman, and Michael O'Neill</i>	

Evolutionary Approaches for Estimating a Coupled Markov Chain Model for Credit Portfolio Risk Management	193
<i>Ronald Hochreiter and David Wozabal</i>	
Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis	203
<i>Piotr Lipinski</i>	
Predicting Turning Points in Financial Markets with Fuzzy-Evolutionary and Neuro-Evolutionary Modeling.....	213
<i>Antonia Azzini, Célia da Costa Pereira, and Andrea G.B. Tettamanzi</i>	
Comparison of Multi-agent Co-operative Co-evolutionary and Evolutionary Algorithms for Multi-objective Portfolio Optimization	223
<i>Rafał Dreżewski, Krystian Obrocki, and Leszek Siwik</i>	
Dynamic High Frequency Trading: A Neuro-Evolutionary Approach	233
<i>Robert Bradley, Anthony Brabazon, and Michael O'Neill</i>	

EvoGAMES Contributions

Decay of Invincible Clusters of Cooperators in the Evolutionary Prisoner's Dilemma Game.....	243
<i>Ching King Chan and Kwok Yip Szeto</i>	
Evolutionary Equilibria Detection in Non-cooperative Games	253
<i>D. Dumitrescu, Rodica Ioana Lung, and Tudor Dan Mihoc</i>	
Coevolution of Competing Agent Species in a Game-Like Environment	263
<i>Telmo Menezes and Ernesto Costa</i>	
Simulation Minus One Makes a Game	273
<i>Noriyuki Amari and Kazuto Tominaga</i>	
Evolving Simple Art-Based Games	283
<i>Simon Colton and Cameron Browne</i>	
Swarming for Games: Immersion in Complex Systems	293
<i>Sebastian von Mammen and Christian Jacob</i>	
Fitness Diversity Parallel Evolution Algorithms in the Turtle Race Game	303
<i>Matthieu Weber, Ville Tirronen, and Ferrante Neri</i>	
Evolving Strategies for Non-player Characters in Unsteady Environments	313
<i>Karsten Weicker and Nicole Weicker</i>	

Grid Coevolution for Adaptive Simulations: Application to the Building of Opening Books in the Game of Go	323
---	-----

*Pierre Audouard, Guillaume Chaslot, Jean-Baptiste Hoock,
Julien Perez, Arpad Rimmel, and Olivier Teytaud*

Evolving Teams of Cooperating Agents for Real-Time Strategy Game	333
--	-----

Paweł Lichocki, Krzysztof Krawiec, and Wojciech Jaśkowski

EvoHOT Contributions

Design Optimization of Radio Frequency Discrete Tuning Varactors	343
---	-----

*Luís Mendes, Eduardo J. Solteiro Pires, Paulo B. de Moura Oliveira,
José A. Tenreiro Machado, Nuno M. Fonseca Ferreira,
João Caldinhas Vaz, and Maria J. Rosário*

An Evolutionary Path Planner for Multiple Robot Arms	353
--	-----

Héctor A. Montes Venegas and J. Raymundo Marcial-Romero

Evolutionary Optimization of Number of Gates in PLA Circuits Implemented in VLSI Circuits	363
---	-----

Adam Slowik and Jacek M. Zurada

Particle Swarm Optimisation as a Hardware-Oriented Meta-heuristic for Image Analysis	369
--	-----

Shahid Mehmood, Stefano Cagnoni, Monica Mordonini, and Muddassar Farooq

EvoIASP Contributions

A Novel GP Approach to Synthesize Vegetation Indices for Soil Erosion Assessment.....	375
---	-----

*Cesar Puente, Gustavo Olague, Stephen V. Smith,
Stephen H. Bullock, Miguel A. González-Botello, and Alejandro Hinojosa-Corona*

Flies Open a Door to SLAM.....	385
--------------------------------	-----

Jean Louchet and Emmanuel Sapin

Genetic Image Network for Image Classification.....	395
---	-----

Shinichi Shirakawa, Shiro Nakayama, and Tomoharu Nagao

Multiple Network CGP for the Classification of Mammograms	405
---	-----

Katharina Völk, Julian F. Miller, and Stephen L. Smith

Evolving Local Descriptor Operators through Genetic Programming....	414
---	-----

Cynthia B. Perez and Gustavo Olague

Evolutionary Optimization for Plasmon-Assisted Lithography	420
<i>Caroline Prodhon, Demetrio Macías, Farouk Yalaoui, Alexandre Vial, and Lionel Amodeo</i>	

An Improved Multi-objective Technique for Fuzzy Clustering with Application to IRS Image Segmentation	426
<i>Indrajit Saha, Ujjwal Maulik, and Sanghamitra Bandyopadhyay</i>	

EvoINTERACTION Contributions

Interactive Evolutionary Evaluation through Spatial Partitioning of Fitness Zones	432
--	-----

Namrata Khemka, Gerald Hushlak, and Christian Jacob

<i>Fractal Evolver</i> : Interactive Evolutionary Design of Fractals with Grid Computing	442
---	-----

Ryan D. Moniz and Christian Jacob

Humorized Computational Intelligence towards User-Adapted Systems with a Sense of Humor	452
--	-----

Pawel Dybala, Michal Ptaszynski, Rafal Rzepka, and Kenji Araki

Innovative Chance Discovery – Extracting Customers' Innovative Concept	462
---	-----

Hsiao-Fang Yang and Mu-Hua Lin

EvoMUSART Contributions

Evolving Approximate Image Filters	467
--	-----

Simon Colton and Pedro Torres

On the Role of Temporary Storage in Interactive Evolution	478
---	-----

Palle Dahlstedt

Habitat: Engineering in a Simulated Audible Ecosystem	488
---	-----

Alan Dorin

The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine	498
--	-----

Arne Eigenfeldt

Filterscape: Energy Recycling in a Creative Ecosystem	508
---	-----

Alice Eldridge and Alan Dorin

Evolved Ricochet Compositions	518
-------------------------------------	-----

Gary Greenfield

Life's What You Make: Niche Construction and Evolutionary Art	528
---	-----

Jon McCormack and Oliver Bown

XXVIII Table of Contents

Global Expectation-Violation as Fitness Function in Evolutionary Composition	538
<i>Tim Murray Browne and Charles Fox</i>	
Composing Using Heterogeneous Cellular Automata	547
<i>Somnuk Phon-Amnuaisuk</i>	
On the Socialization of Evolutionary Art	557
<i>Juan Romero, Penousal Machado, and Antonino Santos</i>	
An Evolutionary Music Composer Algorithm for Bass Harmonization ...	567
<i>Roberto De Prisco and Rocco Zaccagnino</i>	
Generation of Pop-Rock Chord Sequences Using Genetic Algorithms and Variable Neighborhood Search	573
<i>Leonardo Lozano, Andrés L. Medaglia, and Nubia Velasco</i>	
Elevated Pitch: Automated Grammatical Evolution of Short Compositions	579
<i>John Reddin, James McDermott, and Michael O'Neill</i>	
A GA-Based Control Strategy to Create Music with a Chaotic System	585
<i>Costantino Rizzuti, Eleonora Bilotta, and Pietro Pantano</i>	
Teaching Evolutionary Design Systems by Extending “Context Free” ...	591
<i>Rob Saunders and Kazjon Grace</i>	
Artificial Nature: Immersive World Making	597
<i>Graham Wakefield and Haru (Hyunkyung) Ji</i>	
Evolving Indirectly Represented Melodies with Corpus-Based Fitness Evaluation	603
<i>Jacek Wolkowicz, Malcolm Heywood, and Vlado Kesely</i>	
Hearing Thinking	609
<i>Jane Grant, John Matthias, Tim Hodgson, and Eduardo Miranda</i>	

EvoNUM Contributions

Memetic Variation Local Search vs. Life-Time Learning in Electrical Impedance Tomography	615
<i>Jyri Leskinen, Ferrante Neri, and Pekka Neittaanmäki</i>	
Estimating HMM Parameters Using Particle Swarm Optimisation	625
<i>Somnuk Phon-Amnuaisuk</i>	
Modeling Pheromone Dispensers Using Genetic Programming	635
<i>Eva Alfaró-Cid, Anna I. Esparcia-Alcázar, Pilar Moya, Beatriu Femenia-Ferrer, Ken Sharman, and J.J. Merelo</i>	

NK Landscapes Difficulty and Negative Slope Coefficient: How Sampling Influences the Results	645
<i>Leonardo Vanneschi, Sébastien Verel, Marco Tomassini, and Philippe Collard</i>	
On the Parallel Speed-Up of Estimation of Multivariate Normal Algorithm and Evolution Strategies	655
<i>Fabien Teytaud and Olivier Teytaud</i>	
Adaptability of Algorithms for Real-Valued Optimization	665
<i>Mike Preuss</i>	
A Stigmergy-Based Algorithm for Continuous Optimization Tested on Real-Life-Like Environment	675
<i>Peter Korošec and Jurij Šilc</i>	
Stochastic Local Search Techniques with Unimodal Continuous Distributions: A Survey	685
<i>Petr Pošík</i>	
Evolutionary Optimization Guided by Entropy-Based Discretization	695
<i>Guleng Sheri and David W. Corne</i>	
EvoSTOC Contributions	
The Influence of Population and Memory Sizes on the Evolutionary Algorithm's Performance for Dynamic Environments	705
<i>Anabela Simões and Ernesto Costa</i>	
Differential Evolution with Noise Analyzer	715
<i>Andrea Caponio and Ferrante Neri</i>	
An Immune System Based Genetic Algorithm Using Permutation-Based Dualism for Dynamic Traveling Salesman Problems	725
<i>Lili Liu, Dingwei Wang, and Shengxiang Yang</i>	
Dynamic Time-Linkage Problems Revisited	735
<i>Trung Thanh Nguyen and Xin Yao</i>	
The Dynamic Knapsack Problem Revisited: A New Benchmark Problem for Dynamic Combinatorial Optimisation	745
<i>Philipp Rohlfshagen and Xin Yao</i>	
Impact of Frequency and Severity on Non-Stationary Optimization Problems	755
<i>Enrique Alba, Gabriel Luque, and Daniel Arias</i>	
A Critical Look at Dynamic Multi-dimensional Knapsack Problem Generation	762
<i>Sima Uyar and H. Turgut Uyar</i>	

EvoTRANSLOG Contributions

Evolutionary Freight Transportation Planning	768
<i>Thomas Weise, Alexander Podlich, Kai Reinhard, Christian Goroldt, and Kurt Geihs</i>	
An Effective Evolutionary Algorithm for the Cumulative Capacitated Vehicle Routing Problem.....	778
<i>Sandra Ulrich Ngueveu, Christian Prins, and Roberto Wolfler-Calvo</i>	
A Corridor Method-Based Algorithm for the Pre-marshalling Problem	788
<i>Marco Caserta and Stefan Voß</i>	
Comparison of Metaheuristic Approaches for Multi-objective Simulation-Based Optimization in Supply Chain Inventory Management	798
<i>Lionel Amodeo, Christian Prins, and David Ricardo Sánchez</i>	
Heuristic Algorithm for Coordination in Public Transport under Disruptions	808
<i>Ricardo García, Máximo Almodóvar, and Francisco Parreño</i>	
Optimal Co-evolutionary Strategies for the Competitive Maritime Network Design Problem.....	818
<i>Loukas Dimitriou and Antony Stathopoulos</i>	
Author Index	829

Web Application Security through Gene Expression Programming

Jaroslaw Skaruz¹ and Franciszek Seredyński^{2,3}

¹ Institute of Computer Science, University of Podlasie,
Sienkiewicza 51, 08-110 Siedlce, Poland
jaroslaw.skaruz@ap.siedlce.pl

² Polish-Japanese Institute of Information Technology,
Koszykowa 86, 02-008 Warsaw

³ Institute of Computer Science, Polish Academy of Sciences,
Ordona 21, 01-237 Warsaw, Poland
sered@ipipan.waw.pl

Abstract. In the paper we present a novel approach based on applying a modern metaheuristic Gene Expression Programming (GEP) to detecting web application attacks. This class of attacks relates to malicious activity of an intruder against applications, which use a database for storing data. The application uses SQL to retrieve data from the database and web server mechanisms to put them in a web browser. A poor implementation allows an attacker to modify SQL statements originally developed by a programmer, which leads to stealing or modifying data to which the attacker has not privileges. Intrusion detection problem is transformed into classification problem, which the objective is to classify SQL queries between either normal or malicious queries. GEP is used to find a function used for classification of SQL queries. Experimental results are presented on the basis of SQL queries of different length. The findings show that the efficiency of detecting SQL statements representing attacks depends on the length of SQL statements.

Keywords: Intrusion detection, SQL, GEP.

1 Introduction

Nowadays a lot of business applications are deployed in companies to support them with their business activity. These applications are often built with three layer manner: presentation, logical and data. Examples of the data layer are files and the databases containing data while the form of the presentation layer can be a desktop window or a Web site presenting data derived from the data layer and providing application functions. The logical layer is responsible for establishing connection to the database, retrieving data and putting them at the presentation layer. To manage data in the database usually SQL statements are used. When a user executes an application function then SQL query is sent to the database and the result of its execution is shown to the user. Possible security violations exist due to poor implementation of the application. If data provided by the user

are not validated then he can set malicious values of some parameters of SQL query, which leads to the change of the form of the original SQL query. In that case the attacker receives not authorized access to the database. While SQL is used to manage data in the databases, its statements can be ones of sources of events for potential attacks.

The security concern related to business applications aims at ensuring data integrity and confidentiality. One security solution is an intrusion detection system available on the market. It is the application based on attack signatures. When malicious activity matches a signature then an attack is detected. The drawback of this class of security countermeasure is that only those attacks can be detected, which signatures exist. Unfortunately, every a few weeks or even days new security holes are discovered, which allow the attacker to break into the application and steal data. The objective of this work is to build an intelligent system based on GEP, which detects currently known attacks and those, which can occur in the future.

In the literature there are some approaches to intrusion detection in Web applications. In [8] the authors developed anomaly-based system that learns the profiles of the normal database access performed by web-based applications using a number of different models. A profile is a set of the models, to which parts of SQL statement are fed to in order to train the set of the models or to generate an anomaly score. During training phase the models are built based on training data and anomaly score is calculated. For each model, the maximum of anomaly score is stored and used to set an anomaly threshold. During detection phase, for each SQL query anomaly score is calculated. If it exceeds the maximum of anomaly score evaluated during training phase, the query is considered to be anomalous. The number of attacks used in that work was small and the obtained results with the final conclusion should be confirmed.

Besides that work, there are some other works on detecting attacks on a Web server which constitutes a part of infrastructure for Web applications. In [5] a detection system correlates the server-side programs referenced by clients queries with the parameters contained in these queries. It is a similar approach to detection to the previous work. The system analyzes HTTP requests and builds data model based on the attribute length of requests, attribute character distribution, structural inference and attribute order. In a detection phase built model is used for comparing requests of clients.

The paper is organized as follows. The next section discusses SQL attacks. In section 3 we describe GEP. Section 4 shows training data used for experiments. Next, section 5 contains experimental results. Last section summarizes results.

2 SQL Attacks

SQL injection attack consists in such a manipulation of the application communicating with the database, that it allows the user to gain access or to allow it to modify data for which the user has not privileges. To perform an attack in the most cases Web forms are used to inject part of SQL query. Typing SQL keywords and control signs the intruder is able to change the structure of SQL query

developed by a Web designer. If variables used in SQL query are under control of a user, he can modify SQL query which will cause change of its meaning. Consider an example of a poor quality code written in PHP presented below.

```
$connection=mysql_connect();
mysql_select_db("test");
$user=$HTTP_GET_VARS['username'];
$pass=$HTTP_GET_VARS['password'];
$query="select * from users where
    login='$user' and password='$pass'";
$result=mysql_query($query);
if(mysql_num_rows($result)==0)
    echo "authorization failed";
else
    echo "authorization successful"
```

The code is responsible for authorizing users. User data typed in a Web form are assigned to variables *user* and *pass* and then passed to the SQL statement. If retrieved data include one row it means that the user filled in the form login and password the same as stored in the database. Because data sent by a Web form are not validated, the user is free to inject any strings. For example, the intruder can type: ' or 1=1 -- in the login field leaving the password field empty. The structure of the SQL query will be changed as presented below.

```
$query="select * from users where
    login ='' or 1=1 --' and password=''";
```

Two dashes comment the following text. Boolean expression 1=1 is always true and as a result the user will be logged with privileges of the first user stored in the table users.

3 Gene Expression Programming

3.1 Overview

GEP is a modern metaheuristic originally developed by Ferreira [1]. It incorporates ideas of natural evolution derived from genetic algorithm (GA) and evolution of computer programs, which comes from genetic programming (GP) [4]. Since its origination GEP has been extensively studied and applied to many problems such as: time series prediction [6], classification [9][10] and linear regression [2]. GEP evolves a population of computer programs subjected to genetic operators, which leads to population diversity by introducing a new genetic material. GEP incorporates both linear chromosomes of fixed length and expression trees (ET) of different sizes and shapes similar to those in GP. It means that in opposite to the GP genotype and phenotype are separated. All genetic operators are performed on linear chromosomes while ET is used to calculate fitness

of an individual. There is a simple method used for translation from genotype to phenotype and inversely. The advantage of distinction between genotype and phenotype is that after any genetic change of a genome ET is always correct and solution space can be searched through in a more extent.

At the beginning chromosomes are generated randomly. Next, in each iteration of GEP, a linear chromosome is expressed in the form of ET and executed. The fitness value is calculated and termination condition is checked. To preserve the best solution in a current iteration, the best individual goes to the next iteration without modifications. Next, programs are selected to the temporary population, they are subjected to genetic operators with some probability. New individuals in temporary population constitute current population.

3.2 The Architecture of Individuals

The genes of GEP are made of a head and a tail. The head contains elements that represent functions and terminals while the tail can contain only terminals. The length of the head is chosen as a GEP parameter, whereas the length of the tail is calculated according to the eq. 1:

$$\text{tail} = h(n - 1) + 1, \quad (1)$$

where h is the length of the head and n is a number of arguments of the function with more arguments.

Consider an example of a gene presented in eq. 2:

$$+ Qd / + cabdbbca. \quad (2)$$

Its encoded form is represented by ET and shown in figure 1. The length of the gene head presented in eq. 2 equals to 6 and the length of the tail equals to 7 according to eq. 1. The individual shown in figure 1 can be translated to the mathematical expression 3:

$$\sqrt{\frac{a+b}{c}} + d. \quad (3)$$

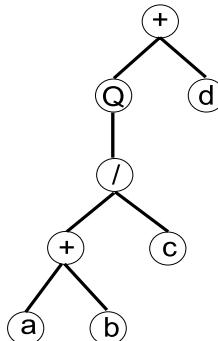


Fig. 1. Expression tree

To construct ET from the linear gene, the analysis must start from the left to the right of the gene elements. The first element of the gene is a root of ET. Next, take such a number of the following elements of the gene that equals to the number of parameters of the function previously taken and put them below it. If a node is a terminal then a branch is completed. If this algorithm of constructing ET is followed it can be seen that some of elements in the tail of the gene do not occur in ET. This is a great advantage of GEP as it is possible to build ET of different sizes and shapes. A genetic change of the gene causes lengthen and shorten of ET.

A chromosome can be built from a few genes. Then sub-ETs are linked by a function - parameter of GEP. For detailed explanation of all genetic operators see [1], [2].

3.3 Fitness Function

In the problem of anomaly detection there are four notions, which allow to look inside performance of the algorithm. True positives (TP) relates to correctly detecting attacks while false positive (FP) means that normal SQL queries are considered as an attack. False negative (FN) concerns attacks as normal SQL queries and true negative (TN) relates to correctly classified normal SQL statements. It is obvious that the larger both TP and TN the better mechanism of classification.

To assess an individual, its fitness should be evaluated. In this work we use sensitivity and precision, which are the most widely used statistics used to describe a diagnostic test [7]. The sensitivity measures proportion of correctly classified attacks and precision refers to the fraction of correctly classified attacks over the number of all SQL queries, which are classified as attacks. Sensitivity and precision are calculated according to eq. 4 and eq. 5:

$$sensitivity = \frac{TP}{TP + FN}, \quad (4)$$

$$precision = \frac{TP}{TP + FP}. \quad (5)$$

Eq. 6 relates to fitness calculation of an GEP individual:

$$fitness = 2 * \frac{precision * sensitivity}{precision + sensitivity}. \quad (6)$$

An individual representing the optimal solution of the problem has fitness equals to 1.0 and the worst chromosome has 0.0. GEP evolve the population of individuals to maximize their fitness value.

4 Training Data

All experiments were conducted using synthetic data collected from a SQL statements generator. The generator takes randomly a keyword from selected subset

of SQL keywords, data types and mathematical operators to build a valid SQL query. Since the generator was developed on the basis of the grammar of the SQL language, each generated SQL query is correct. We generated 3000000 SQL statements. Next, the identical statements were deleted. Finally, our data set contained thousands of free of attack SQL queries. The set of all SQL queries was divided into 20 subsets (instances of the problem), each containing SQL statements of different length, in the range from 10 to 29 tokens (see below). Data with attacks were produced in the similar way to that without attacks. Using available knowledge about SQL attacks, we defined their characteristic parts. Next, these parts of SQL queries were inserted randomly to the generated query in such a way that it provides grammatical correctness of these new statements. Queries in each instance were divided into two parts: for training GEP and testing it. Each of the part contains 500 SQL statements.

Classification of SQL statements is performed on the basis of their structure. Each SQL query is divided on distinct parts, which we further call tokens. In this work, the following tokens are considered: keywords of SQL language, numbers and strings. We used the collection of SQL statements to define 36 distinct tokens. The table 1 shows selected tokens, their indexes and the coding real values.

Table 1. A part of a list of tokens and their coding values

token	index	coding value
SELECT	1	0.1222
FROM	2	0.1444
...
...
UPDATE	9	0.3
...
number	35	0.8777
string	36	0.9

Each token has assigned the real number. The range of these numbers starts with 0.1 and ends with 0.9. The values assigned for the tokens are calculated according to eq. 7:

$$\text{coding vallue} = 0.1 + (0.9 - 0.1)/n * k, \quad (7)$$

where n is the number of all tokens and k is the index of each token.

Below, there is an example of a SQL query:

SELECT name FROM users (8)

To translate the SQL query the table 1 is searched through to meet a token. The first token of the SQL query shown in eq. 8 is *SELECT* and the corresponding coding value equals to 0.1222. This step is repeated until all tokens are translated. Finally a vector: 0.1222, 0.9, 0.1444, 0.9 is received as an encoded form of query represented by eq. 8. All elements of the vector are terminals used to generating individuals of GEP.

5 Experimental Results

In this section we are going to use GEP to find a function that can be used to classify SQL statements. Each of twenty instances of the problem consists of two parts: training and testing data. Each time GEP was run first on training part of an instance of the problem and next found classification rule was tested on the second part of the instance. In this work we apply in the most cases the same values of parameters as in [3]. As the search space is bigger than this in [3], the number of individuals was increased to 100.

In this classification problem very simple set of functions was chosen, which consists of arithmetic operators ($+$, $-$, $*$, $/$). The set of terminals depends on the length of SQL queries used for training. The number of terminals equals to the number of tokens, which constitute SQL query. As a selection operator roulette wheel was chosen. Figure 2 shows detection system performance during training and testing.

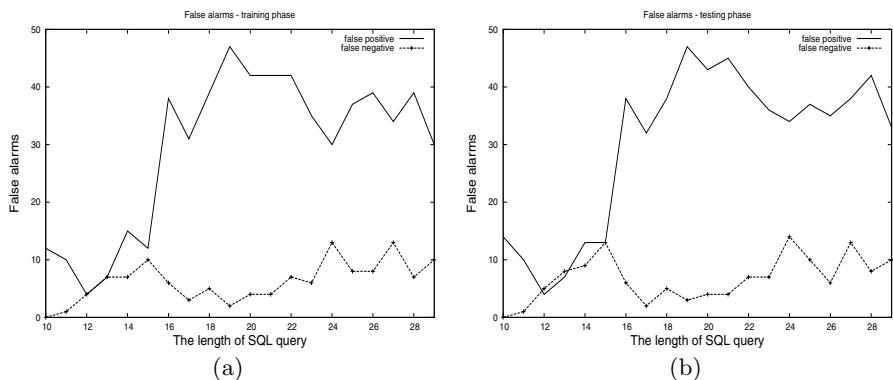


Fig. 2. GEP performance, training phase (a), testing phase (b)

It is easily noticeable that the false alarms rate in both figures 2 a) and b) are very similar. Presented percentage values of false alarm rate are averaged over 10 runs of GEP. Such results of the experiment allow to say that the best evolved mathematical expression classifies SQL queries in the testing set with nearly the same efficiency as in the training set. One of the reasons this happens is that although SQL statements are placed randomly to both data sets, they features similar structure in both data sets, which were used in the classification task. From the figure 2 b) it can be seen that the false negative rate changes in small extent for SQL queries of different length. The averaged FN over SQL queries with various number of tokens equals to 6.85 and the standard deviation equals to 2.6. At the same time the averaged FP rate equals to 33.43 with the standard deviation equals to 14.54. SQL statements constituted from 10 to 15 tokens are classified with lower error than longer SQL queries. For these shorter queries, the averaged sum of false alarms for each length of SQL equals to 17.4%. Figure 3 shows fitness of the best individual for SQL queries consisting of 10, 20 and 29

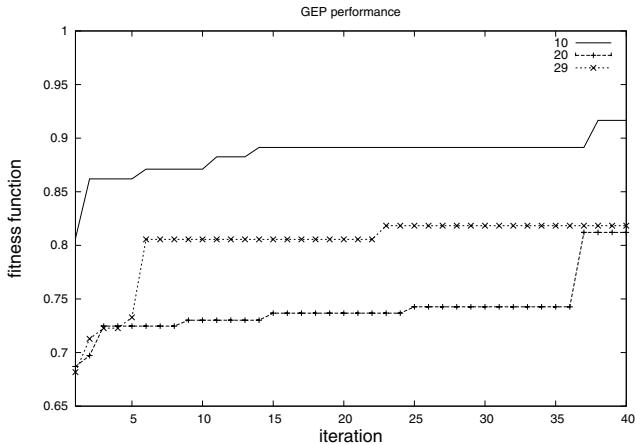


Fig. 3. Algorithm run for SQL queries made of 10, 20 and 29 tokens

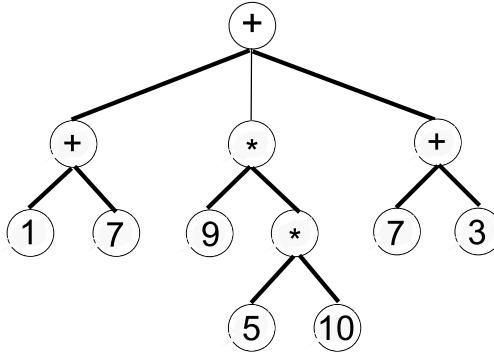


Fig. 4. The form of the classification rule evolution

tokens. The findings show a great ability of GEP to find a good solution. At the beginning of the algorithm better solutions are found quickly. Next, they are improved a few times. The best fitness is obtained for SQL queries made of 10 tokens. The longer SQL statements are classified with nearly the same extent, which is confirmed by the charts in 40 generation for queries with 20 and 29 tokens. The form of the best individual for the instance with SQL statements constituted from 10 tokens is shown in figure 4. The numbers in the expression tree represent positions of tokens within SQL statements. ET transformed to the mathematical expression is presented in eq. 9:

$$1 + 7 + (9 * 5 * 10) + 7 + 3. \quad (9)$$

The classification function for the instance with 10 tokens was discovered in 36 iteration of GEP. Figure 5 shows the number of various tokens used for constructing the classification rule during each iteration of GEP. The number

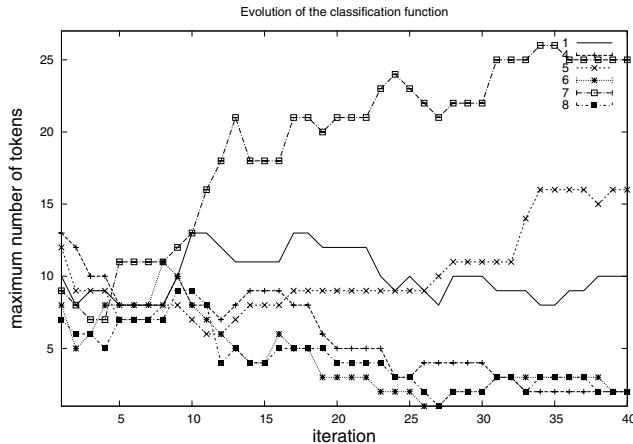


Fig. 5. The form of the classification rule evolution

of tokens was calculated among the best 20 individuals. In the first iteration of GEP the initial number of tokens contained in 20 individuals is nearly the same. During evolutionary process some tokens, which make the classification function more powerful, occur more often in the individuals. At the same time there are also some unuseful tokens, which does not allow classify SQL queries. Tokens at position 8, 6 and 4 within SQL queries are not needed for classification of SQL queries made of 10 tokens and their existence in the individuals decreases in the next iterations of GEP.

6 Conclusions

In the paper we have presented an application of modern evolutionary meta-heuristic to the problem of detecting intruders in Web applications. We shown typical SQL attack and transformation the problem of anomaly detection to classification problem. Classification accuracy depicts great efficiency for SQL queries constituted from 10 to 15 tokens. For longer statements the averaged FP and FN equals to about 23%. We have also presented dynamics of GEP, which reveals some important features. On the one hand minimal change in genotype leads to a great change in phenotype. On the other hand search space is searched through in more extent. We believe that it is possible to keep the advantages of GEP and at the same time to make ETs less susceptible to great changes.

Acknowledgment

This work was supported by the Ministry of Science and Higher Education under grant no. N N519 319935.

References

1. Ferreira, C.: Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Systems* 13(2), 87–129 (2001)
2. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Angra do Heroísmo, Portugal (2002)
3. Ferreira, C.: Gene Expression Programming and the Evolution of Computer Programs. In: de Castro, L.N., Von Zuben, F.J. (eds.) *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches (Recent Developments in Biologically Inspired Computing)*. Idea Group Publishing (2004)
4. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
5. Kruegel, C., Vigna, G.: Anomaly Detection of Web-based Attacks. In: Proc. 10th ACM Conference on Computer and Communication Security, pp. 251–261 (2003)
6. Litvinenko, V.I., Bidyuk, P.I., Bardachov, J.N., Sherstjuk, V.G., Fefelov, A.A.: Combining Clonal Selection Algorithm and Gene Expression Programming for Time Series Prediction. In: Proc. Third Workshop 2005 IEEE : Technology and Intelligent Data Acquisition and Advanced Computing Systems Applications, pp. 133–138 (2005)
7. Linn, S.: A New Conceptual Approach to Teaching the Interpretation of Clinical Tests. *Journal of Statistics Education* 12(3) (2004)
8. Valeur, F., Mutz, D., Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks. In: Proc. Conference on Detection of Intrusions and Malware and Vulnerability Assessment, Austria (2005)
9. Zhou, C., Nelson, P.C., Xiao, W., Tirpak, T.M.: Discovery of Classification Rules by Using Gene Expression Programming. In: Proc. International Conference on Artificial Intelligence, Las Vegas, pp. 1355–1361 (2002)
10. Zhou, C., Xiao, W., Nelson, P.C., Tirpak, T.M.: Evolving Accurate and Compact Classification Rules with Gene Expression Programming. *IEEE Transactions on Evolutionary Computation* 7(6), 519–531 (2003)

Location Discovery in Wireless Sensor Networks Using a Two-Stage Simulated Annealing

Guillermo Molina and Enrique Alba

Departamento de Lenguajes y Ciencias de la Computación
University of Málaga, 29071 Málaga, Spain
`{guillermo,eat}@lcc.uma.es`

Abstract. Wireless Sensor Networks (WSN) monitor the physical world using small wireless devices known as sensor nodes. Location information plays a critical role in many of the applications where WSN are used. A widely used self-locating mechanism consists in equipping a small subset of the nodes with GPS hardware, while the rest of the nodes employ reference estimations (received signal strength, time of arrival, etc.) in order to determine their locations. Finding the location of nodes using node-to-node distances combined with a set of known node locations is referred to as Location Discovery (LD). The main difficulty found in LD is the presence of measurement errors, which results in location errors. We describe in this work an error model for the estimations, propose a two-stage Simulated Annealing to solve the LD problem using this model, and discuss the results obtained. We will put a special stress on the improvements obtained by using our proposed technique.

1 Introduction

One key feature in a Wireless Sensor Network (WSN) is the location or context awareness of the gathered information. Many WSN applications, like as target acquisition [1], surveillance [2], or a forest fire prevention [3], require geographical information associated with the retrieved data. If action needs to be taken in response to a detected event (enemy troops moving, fire, etc.), it is essential to know *where* that event is located.

A simple way to ensure this is to equip every node in the WSN with GPS. However, GPS hardware is expensive, whereas sensor nodes are generally cost-constrained since the WSN will likely contain a very high amount of nodes. Therefore only a small subset of the network can affordably be equipped with GPS; these nodes are called *anchor* nodes, or beacons. A localization process is required for the rest of the nodes in the network, using these anchor nodes as references. Distance or angle measurements between sensor nodes are employed, and a multilateration system can be defined. The process of determining the nodes locations in a WSN is called Location Discovery (LD).

As will be explained in detail further, the measurements available for the nodes in a WSN usually contain errors. These errors range from slight errors to large ones, and are difficult to model by a standard model such as Gaussian. These

distance measurement errors may result in severe location errors that degrade the WSN performance, and therefore should be taken into account during the LD. We will present in this paper a model for the measurement errors and later use it in a Simulated Annealing optimization algorithm to solve real-world instances of LD. We will analyze two of the most popular fitness functions for LD: the error norm function L_1 and the Likelihood function. Our contribution with this work is to combine the use of these two functions in a Simulated Annealing in a way that outperforms the results obtained with each one separately.

The rest of the paper is organized as follows. In Section 2 the Location Discovery problem is explained and formulated, and the distance error model is presented. The fitness functions are discussed in Section 3, and the proposed optimization technique is presented in Section 4. Then in Section 5 the experiments performed and the results obtained are shown. Finally, the conclusions are drawn in the last section.

2 Location Discovery

In this section we first present the basic formulation of the LD problem, and then introduce the error models used in this work.

2.1 Problem Formulation

The Location Discovery problem is an optimization problem where a set of element locations has to be determined in a manner such that the location error is minimized. It is widely considered one of the fundamental tasks in a WSN [4]. The LD affects many aspects of a sensor networks, from application (context-aware monitoring) to communication protocols (geographical routing [5]). The location discovery problem has been proven to be NP-complete [6].

Let a and b be two points whose coordinates are (x_a, y_a) and (x_b, y_b) respectively, and (x'_a, y'_a) and (x'_b, y'_b) their estimated coordinates (aka estimated locations). We define the following:

- Real distance $d_{a,b} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$
- Measured or estimated distance $\delta_{a,b}$
- Calculated distance $c_{a,b} = \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2}$
- Measurement error $\epsilon_{a,b} = \delta_{a,b} - d_{a,b}$
- Calculated error $\epsilon'_{a,b} = c_{a,b} - d_{a,b}$
- Location errors $\epsilon_a = \sqrt{(x_a - x'_a)^2 + (y_a - y'_a)^2}$, $\epsilon_b = \sqrt{(x_b - x'_b)^2 + (y_b - y'_b)^2}$

Our formulation of the problem is as follows: given a set S of N nodes s_i , $1 \leq i \leq N$, a subset of nodes s_j , $1 \leq j \leq K < N$ with previously known locations (anchor nodes), and a set of distance estimations $\delta_{i,j}$, $1 \leq i, j \leq N, i \neq j$, we have to determine the locations of every node s_l , $K < l \leq N$ that minimize the location error.

When there are no measurement errors (i.e. measured distances equal real distances, $\delta_{i,j} = d_{i,j}$), the location discovery can be solved to optimality. In this

scenario, finding the locations for which the calculated distances are equal to the measured ones will solve the problem.

However, in a real system there are significant distance measurement errors. In this scenario, the solution that produces calculated distances closest to the measured ones is not necessarily the optimum, since the real objective in LD is to minimize the *location error*, and not the *measurement error*. This is one of the main difficulties of this problem, precisely because we are trying to minimize a value (the location error) that is impossible to determine (otherwise we would already know the optimal solution to the problem).

A technique that solves LD as if there were no measurement errors will have limited location accuracy. But if a model of the distance measurement error is employed, the quality of the results obtained can be greatly improved.

2.2 Distance Error Model

The data we employ was gathered from several experiments performed at the Fort Leonard Wood Self Healing Minefield Test Facility [7]. Those experiments deployed a WSN containing from 79 to 93 sensor nodes, which are custom design on an SH4 microprocessor running at 200 MHz. The nodes are equipped with four independent speakers and microphones and use ToA on the acoustic signal to determine the distance between themselves [8].

In total, there are 33 sets of distance measurements collected over the course of a few days. Each set consists of a single round of acoustic signal transmission by all the nodes. Figure 1 (left) shows a graphical representation containing all the data from the 33 sets. The complete data contains information about more than 20,000 distance measurements. Each measurement is represented by a point whose abscisse is the real distance between the nodes $d_{i,j}$ (which is known in this case) and the ordinate is the distance measured by the nodes $\delta_{i,j}$. The WSN was deployed on an area of 200m × 50m.

This data is used to set up a non-parametric probability distribution function for the link distances, by using a kernel function as in [9]. Figure 1 (right)

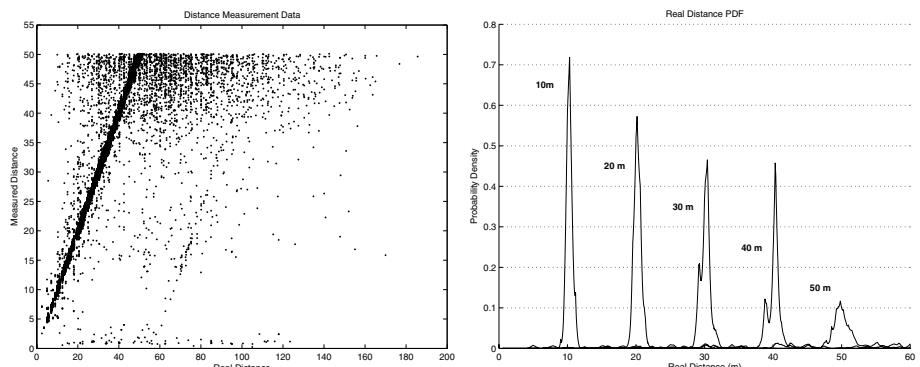


Fig. 1. Distance measurements error model: measurement data (left), and kernel probability density function (right)

shows the (superimposed) probability density function (pdf) obtained for five different values of measured link distance: 10, 20, 30, 40 and 50 meters. For testing purposes, we select 10 data sets that constitute the problem instances, while the remaining 23 serve as the reference data to establish the model.

This model, when properly introduced into the guiding fitness function of the search algorithm, will improve the accuracy of the results obtained.

3 Fitness Function

There are two fitness functions that have been used to solve LD. The first fitness function is an error norm function, and the second is a likelihood function. In this section we present both functions and explain why the two are used.

3.1 Norm Function

The first fitness functions is an error norm function, in which the total magnitude of the error is calculated, and the resulting output has to be minimized.

Typical norms are L_1 , L_2 or L_∞ [10,11] (equations 1, 2, and 3).

$$L_1 = |\epsilon_1| + |\epsilon_2| + |\epsilon_3| + \dots + |\epsilon_L| , \quad (1)$$

$$L_2 = \sqrt{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \dots + \epsilon_L^2} , \quad (2)$$

$$L_\infty = \max\{\epsilon_1, \epsilon_2, \epsilon_3, \dots, \epsilon_L\} , \quad (3)$$

where we have the measurement errors $\epsilon_i = c_{i1,i2} - \delta_{i1,i2}$.

In this work, we use the L_1 norm function, with weighted error values (Equation 4). The weight w of an link distance error, which ranges from 0 to 1, indicates the confidence that this error is significant, which correspond to the confidence on the involved distance measurements.

$$L_1 = w_1|\epsilon_1| + w_2|\epsilon_2| + w_3|\epsilon_3| + \dots + w_L|\epsilon_L| . \quad (4)$$

This confidence values are generated from the knowledge of the measurement errors (Section 2.2). We employ in this work a binary weighting function, that only discriminates reliable measurements from unreliable ones (Equation 5).

$$w(\delta) = \begin{cases} 1 & \text{if } 5 \leq \delta \leq 40 , \\ 0 & \text{else} \end{cases} . \quad (5)$$

3.2 Likelihood Function

The second approach is to solve the LD as a Maximum Likelihood optimization problem (ML). In this approach, the distance measurement error model (described in Section 2.2) is used to calculated the *likelihood* (i.e. probability) of any give solution candidate. For every pair (c_{ij}, δ_{ij}) of link distances, a probability density $P(c_{ij}, \delta_{ij})$ can be calculated with the kernel distance error model.

Then, all probabilities P are multiplied and the resulting output, called the likelihood L , is subject to maximization (Equation 6).

$$L = P(d_1, \delta_1) * P(d_2, \delta_2) * P(d_3, \delta_3) * \dots * P(d_L, \delta_L). \quad (6)$$

We add a couple of modifications to the likelihood calculation. First, since we found experimentally that there is almost always some link distance that will produce $P(c, \delta) = 0$, a floor value is added to every single probability to avoid the annulation of the whole likelihood. After some experimentation this value was set to 10^{-6} . Second, since the range of possible values of L is extremely large, and we want our search technique to keep a constant pressure on the solution improvement regardless of the current value of L , instead of the linear value, we use the logarithm of L as the fitness value.

3.3 Fitness Consistency

Let s_a and s_b be two possible solutions for a given LD instance whose optimal solution is known, and $LE()$ be the location error function. Then $LE(s_a) < LE(s_b)$ means s_a outperforms s_b . When this is reflected by the fitness function, we say the fitness function is *consistent* for this pair of nodes.

We define two scenarios: the first scenario consists of a pool of randomly generated pairs of solutions (presumably low quality solutions); the second scenario consists of pairs of random solutions with low average location error (high quality solutions). For this second scenario three different location error levels have been used: 1m, 10cm and 1cm.

For each scenario we generate 10,000 random pairs of solutions, and check the maximum likelihood, the L_1 and L_∞ norm functions. The percentages of consistency are shown in Table 1.

Table 1. Consistency of the Maximum Likelihood, L_1 and L_∞ norm functions for different location errors (%)

Scenario	ML	L_1	L_∞
Pure random	55.8	62.1	54.7
Loc. Error 1 m	69.1	66.8	49.4
Loc. Error 10 cm	68.5	55.8	50.4
Loc. Error 1 cm	71.7	51.9	50.1

We notice that L_1 has higher consistency than ML for random solutions (62.1% vs 55.8%), but the situation reverses when solutions get close to the optimum; the closer they are to the optimum, the more ML outperforms L_1 (66.8% vs 69.1%, 68.5% vs 55.8% and 71.7% vs 51.9% for errors of 1m, 10cm and 1cm respectively). The L_∞ norm is virtually undistinguishable from random search (50% consistency), and therefore should not be employed in this problem.

As a consequence, we state that the L_1 norm function is preferable for the beginning of the search process (starting from a random solution), while ML is preferable for the end of the search process.

4 Optimization Algorithm

In this section we present the search algorithms we propose to solve location discovery. These algorithms handle one solution or a population of solution, and employ a fitness function (described in the previous section) to guide the search.

4.1 Canonical SA

Simulated annealing is a trajectory based optimization technique [12]. It was first proposed by Kirkpatrick et al. in [13]. The pseudocode for this algorithm is shown in Algorithm 1.

Algorithm 1. Pseudocode of SA

```

 $t \leftarrow 0;$ 
Initialize( $T, s_a$ )
Evaluate( $s_a$ )
while not EndCondition( $t, s_a$ ) do
    while not CoolingCondition( $t$ ) do
         $s_n \leftarrow \text{ChooseNeighbor}(s_a)$ 
        Evaluate( $s_n$ )
        if Accept( $s_a, s_n, T$ ) then
             $s_a \leftarrow s_n$ 
        end if
         $t \leftarrow t + 1$ 
    end while
    Cooldown( $T$ )
end while

```

The algorithm works iteratively keeping a single tentative solution s_a at any time. In every iteration, a neighbor solution s_n is generated, which either replaces not the current solution depending on an acceptance criterion. The acceptance criterion works as follows: both the old (s_a) and the new (s_n) solutions have associated quality values (*fitnesses*); if the new solution has better fitness than the current solution, it replaces the current solution. Otherwise, the replacement is done with probability P , which depends on the difference between their quality values and a control parameter T (*temperature*). This acceptance criterion provides a way of escaping from local optima. The mathematical expression for the probability P is shown in Equation 7.

$$P = \frac{2}{1 + e^{\frac{\text{fitness}(s_a) - \text{fitness}(s_n)}{T}}} \quad (7)$$

The temperature parameter is reduced following a given cooling schedule. In this work we employ the geometric rule $T(n+1) = \alpha \cdot T(n)$, with $0 < \alpha < 1$, performed every k iterations (k is the *Markov chain length*). This makes SA accept only better solutions towards the end of the search.

A mutation operator is used to produce s_n from s_a as follows. Every node location in s_a is modified with a given probability p . When a location is modified,

a translation vector is added to the current coordinates; the translation vector is generated with random angle θ and random lenght r between 0 and a given range. After some experimentation, the range was set to 5 times and 1 time the average link error for the first and second stages respectively, where the average link error is calculated as L_1/L (L_1 is the error norm and L is the number of links measured).

4.2 Two-Stage Resolution

As was shown in Section 3.3, the L_1 norm provides a low accuracy near the optimum, but also a good guidance for solutions far from the optimum. With ML, we obtain a higher consistency in the neighborhood of the optimum, but a poor guidance for far away solutions.

Therefore, we propose a two-stage solving process that combines the two fitness functions. The basic intuition is to use each function where it performs best. Thus, in the first phase L_1 is used starting from a random initial solution, then in the second phase the output is refined using ML. More details on this can be found in Section 5.

5 Experiments and Results

We selected 10 measurement sets as problem instances, and compiled the error model from the 23 remaining sets, so that no information from an instance is used in the process of resolving it. The number of beacons is set to 10% of the total number of nodes, the beacons are randomly selected for each run. Figure 2 shows an example of a solution with an average location error of 1.5m and 13m for a given problem instance (3-19F). The dots represent the nodes real locations, the asteriscs represent the estimated locations and a dotted links both. In the following, our analysis and comments will be based on the location error results as a measure of the solution quality.

The starting solution is randomly generated. Every problem instance is solved using information from a single round of distance measurements (he results are expected to improve if several measurement rounds are available). Each test case is solved 30 independent times, each independent execution is stopped after performing 5,000,000 solution evaluations. The algorithm parameters are empirically tuned, the values obtained are displayed in Table 2.

We first solved the instances using either the error norm minimization or the maximum likelihood, then employed the two-stage resolution process in order to measure the improvement obtained by merging the two phases into a single search process.

5.1 Stand-Alone Phases

We first used each solving phase on its own, starting from a random solution. The results obtained are shown in columns two to five of Table 3. For each of the

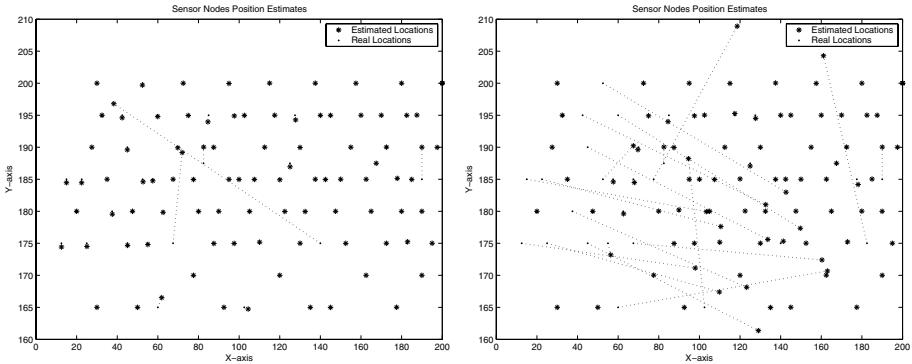


Fig. 2. Example solutions: Average location errors 1.5m (left), and 13m (right)

Table 2. Parameter values

Parameter	Only Phase I	Only Phase II	Two-stage	
			Phase I	Phase II
Evaluations	5,000,000	5,000,000	3,000,000	2,000,000
Mutation rate (%)	4	4	4	4
Initial T	150	10	150	0.5
Markov chain length	50	50	50	50
α	0.99995	0.99995	0.99995	0.9999

solving procedures we show the average fitness obtained (minimum error norm and maximum likelihood respectively), and the average location errors obtained.

From the results in Table 3, the phase II search gets better results than the phase I overall. However, the average location error obtained using phase I has a tighter upper bound, which means that phase II will occasionally obtain much larger location errors (as in instance 3-19C). A more detailed observation on the results confirms it. Phase I, which uses the error norm function as fitness function obtains a location error generally ranging from 1 to 10 meters. Phase II, which uses the maximum likelihood criterion, gets solutions that are either highly accurate (approximately 10 cm location error), or grossly inaccurate (from 10 to 30 m). This fits our intuition from Section 3.3.

5.2 Two-Stage Resolution

Table 3 shows the results obtained by our proposed two-stage optimization process. Columns six and seven show the average fitness (maximum likelihood) and location error respectively, while the last two columns show the variation of the location error (in percentage) between phase I and the two-stage resolution (column eight), and between phase II and the two-stage resolution (column nine), computed with respect to the largest location error value.

With the two-stage solving process, we either reach high accuracy solutions, or obtain a moderate location error (from 1 to 10 meters). Thus, our two-stage optimization technique effectively combines the best features of the two individual solving processes. Compared to the separate processes, our proposed technique

Table 3. Average fitness and location errors(m)

Problem Instance	Phase I		Phase II		Two-Stage		Comparisons %	
	fitness	loc. error	fitness	loc. error	fitness	loc. error	phase I	phase II
3-19A	3135.19	8.185	268.36	3.016	347.12	1.250	-84.72	-58.55
3-19B	2808.53	4.544	228.94	2.109	280.46	2.215	-51.25	+5.20
3-19C	873.02	4.838	86.26	11.075	230.25	4.770	-1.41	-56.93
3-19D	2062.91	5.305	258.77	4.909	279.39	7.188	+26.20	+31.65
3-19E	636.757	5.778	150.77	8.606	222.59	4.965	-14.07	-42.31
3-19F	1723.42	5.033	276.04	3.826	388.79	1.851	-63.22	-51.62
3-20A	3685.64	5.382	317.39	4.736	391.20	3.739	-30.52	-21.06
3-20B	3146.39	2.033	512.13	1.113	630.86	0.809	-60.21	-27.33
3-25A	2915.29	3.692	581.35	2.011	703.93	0.146	-96.05	-92.74
3-25B	6676.39	3.306	572.87	0.777	207.11	6.275	+47.30	+87.60

obtains the best results in eight out of the ten instances solved when compared with phase I, and in seven instances when compared with phase II; overall, it obtains the best results for seven instances. It reduces the location error obtained with phase I by more than 30% in six instances, and in five instances compared to phase II, with two other instances having an improvement over 20%. On the negative side, for instances 3-19D and 3-25B the two-stage process is largely outperformed by both single phases. It appears that for these instances, the phase II performs best; the poor results obtained with the proposed method might be caused by a poor balance between the two stages; increasing the weight of the second phase might improve the results. Therefore, we conclude that the two-stage search process constitutes an interesting alternative to the archetypical error norm minimization and maximum likelihood criteria, outperforming both of them in 70% of our conducted experiments.

6 Conclusions

We addressed in this work one of the fundamental problems of the Wireless Sensor Network (WSN) field, the Location Discovery problem (LD). The objective in this problem is to determine the locations of a set of sensor nodes given a subset of node locations (beacons) and a list of measured node-to-node distances. The difficulty of this NP-hard problem increases when the distances measured contain errors. We propose an optimization technique that uses a distance error model and a two-stage solving process to solve the LD problem. This solving process uses an error-norm minimization process to get a rough initial guess, then polishes this solution using a maximum-likelihood optimization. This combined technique obtains highly accurate results and proves its worth by outperforming both of the separate optimization stages.

As future work we plan to use other search algorithms with a clear focus on the robustness, since we observed an irregular behavior of the solving algorithm for the different instances (poor results for instances 3-19D and 3-25B). We also plan to further study the switching condition between the two search phases.

Acknowledgements

Authors are partially funded by the Andalusian Government under contract P07-TIC-03044 (The DIRICOM project, <http://diricom.lcc.uma.es/>), and by grant AP2005-0914 from the Spanish government.

References

1. Nemeroff, J., Garcia, L., Hampel, D., DiPierro, S.: Application of sensor network communications. In: Military Communications Conference (MILCOM) 2001. Communications for Network-Centric Operations: Creating the Information Force, vol. 1, pp. 336–341. IEEE, Los Alamitos (2001)
2. Lédeczi, Á., Nádas, A., Völgyesi, P., Balogh, G., Kusy, B., Sallai, J., Pap, G., Dóra, S., ároly Molnár, K., Maróti, M., Simon, G.: Countersniper system for urban warfare. *ACM Trans. Sen. Netw.* 1(2), 153–177 (2005)
3. Mladineo, N., Knezic, S.: Optimisation of forest fire sensor network using gis technology. In: Proceedings of the 22nd International Conference on Information Technology Interfaces (ITI), pp. 391–396 (2000)
4. Koushanfar, F., Slijepcevic, S., Potkonjak, M., Sangiovanni-Vincentelli, A.: Location discovery in ad-hoc wireless sensor networks. In: Cheng, X., Huang, X., Du, D.Z. (eds.) *Ad Hoc Wireless Networking*, pp. 137–173. Kluwer Acad. Publish., Dordrecht (2003)
5. Karp, B., Kung, H.T.: Gpsr: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th annual international conference on Mobile computing and networking (MOBICOM), pp. 243–254. ACM Press, New York (2000)
6. Moore, D., Leonard, J., Rus, D., Teller, S.: Robust distributed network localization with noisy range measurements. In: Proceedings of the 2nd int. conf. on Embedded networked sensor systems (SenSys), pp. 50–61. ACM, New York (2004)
7. Merrill, W., Newberg, F., Girod, L., Sohrabi, K.: Battlefield ad-hoc lans: a distributed processing perspective. *GOMACTech.* (2004)
8. Girod, L.: Development and characterization of an acoustic rangefinder (2000)
9. Feng, J., Girod, L., Potkonjak, M.: Location discovery using data-driven statistical error modeling. In: Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM), pp. 1–14 (2006)
10. Koushanfar, F., Slijepcevic, S., Wong, J., Potkonjak, M.: Global error-tolerant algorithms for location discovery in ad-hoc wireless networks. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 4, Proceedings, pp. IV–4186 (2002)
11. Slijepcevic, S., Megerian, S., Potkonjak, M.: Location errors in wireless embedded sensor networks: sources, models, and effects on applications. *SIGMOBILE Mob. Comput. Commun. Rev.* 6(3), 67–78 (2002)
12. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys* 35(3), 268–308 (2003)
13. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* 4598(220), 671–680 (1983)

Wireless Communications for Distributed Navigation in Robot Swarms

Gianni A. Di Caro, Frederick Ducatelle, and Luca M. Gambardella*

“Dalle Molle” Institute for Artificial Intelligence Studies (IDSIA)

Galleria 2, 6928 Manno, Switzerland

{gianni,frederick,luca}@idsia.ch

Abstract. We consider a swarm of robots equipped with an infrared range and bearing device that is able both to make estimates of the relative distance and angle between two robots in line-of-sight and to transfer data between them. Through the infrared range and bearing device, the robots create a line-of-sight mobile ad hoc network. We investigate different ways to implement a swarm-level distributed navigation function exploiting the routing information gathered within this network. In the scenario we consider, a number of different events present themselves in different locations. To be serviced, each event requires that a robot with the appropriate skills comes to its location. We present two swarm-level solutions for guiding the navigation of the selected robots towards the events. We use a bio-inspired ad hoc network routing protocol to dynamically find and maintain paths between a robot and an event location in the mobile line-of-sight network, and use them to guide the robot to its goal. The performance of the two approaches is studied in a number of network scenarios presenting different density, mobility, and bandwidth availability.

1 Introduction

In parallel with recent and continuous advances in the domain of mobile ad hoc networks (MANETs), the field of *networked robotics* is attracting an increasing interest. The idea is to equip teams or swarms of robots with wireless communication devices and allow them to exchange information to support cooperative activities and fully exploit ensemble capabilities. In this work, we use networked robotics in a situation where a swarm of robots needs to execute tasks in an indoor area. The term “swarm” refers here to a potentially large group of small robots that collaborate using weak coordination mechanisms [8]. The tasks correspond to events that need to be serviced in given locations. Each event can be taken care of by a single robot able to provide a specific set of functionalities. A full solution to this problem involves mechanisms for announcing events, for the allocation of robots to events and for guiding robots to event locations. Here

* This work was supported by the SWARMANOID project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission under grant IST-022888.

we focus on robot navigation: how can a robot find an event's location after the event has been advertised and the robot has been selected. An important aspect of the problem under study is the fact that the robots cooperatively and transparently help each other for navigation while they are at the same time involved in a task of their own. This is different from most of previous works, where all robots are involved in solving a single task cooperatively (e.g. collaboratively guide one robot to a destination [12,11]).

The robots our work is based on are the *foot-bots* [2], which are small mobile robots developed within the *Swarmanoid* project [1]. They move around on a combination of tracks and wheels, and are equipped with three different devices for wireless communication: Wi-Fi, Bluetooth, and an *infrared range and bearing system (Ir-RB)*. The latter one consists of a number of infrared transmitters and receivers placed all around the robot. The system allows both the wireless transmission of data over short distances along *line-of-sight (LoS) paths*, as well as the estimation of the relative *distance* and *angle* to other robots. We make use of the Ir-RB system to create a LoS MANET between the robots of the swarm (in parallel, the Wi-Fi MANET can be used for announcing and allocating the events as well as to transmit the geographic data regarding LoS paths if the Ir-RB bandwidth is insufficient). The core idea to implement a distributed navigation function is to set up a route over the LoS MANET between a robot that wants to serve an event and the event's location. Since each robot in the swarm is involved in its own task, this MANET presents frequent and unexpected topology changes. Therefore, we rely on a bio-inspired adaptive routing algorithm, called *AntHocNet*, which is able to find and maintain routes in the face of high network mobility and has been shown to give efficient and robust performance also in large networks and in cluttered environments [6,4]. The established route is used for robot navigation. We distinguish two modes of operation: in the first the robot physically follows (robot by robot) the route formed via the wireless connections, while in the other the route is used to make estimates of the relative position of the event, so that the robot can aim to go there independently of the actual data route.

The proposed system has some important advantages. First, thanks to the use of the Ir-RB system, robots can get relative positioning information without a central reference system such as GPS. Second, since robots transparently guide each other via wireless communication and are supported by an adaptive routing algorithm, they do not need to adapt their movements as is done in other approaches (e.g., based on visual communication [11]). At the same time, they can get involved in tasks of their own, improving the possibilities for parallel task solving. Finally, the possibility to base robot navigation on an estimate of the relative location of the event to be serviced, and independent from the actual MANET route, allows to overcome moments of interrupted network connectivity.

This paper is organized as follows. In Section 2, we describe the robots for which we developed this work. Then, in Section 3, we present our communication aided robot navigation system. After that, in Section 4 we present and discuss the results of our experiments.

2 The Robots and Their Communication Interfaces

A foot-bot is about 15 cm wide and long and 20 cm high. It moves on the ground making use of a combination of tracks and wheels. For *basic obstacle avoidance and navigation*, the foot-bot has 24 short-range infrared sensors all around, 8 infrared sensors directed at the floor, and a rotating platform with a long-range infrared sensor (maximum range 150 cm, precision 2 cm), which gives one measurement per second of obstacles all around with a step size of 2° . For *communication*, the foot-bot has Wi-Fi and Bluetooth as well as the mentioned Ir-RB system, which is made of 26 infrared emitters and 16 receivers, placed around and on top of the foot-bot. Based on the quality of the received signals, it calculates an estimate of the relative bearing and range to other robots in LoS equipped with the same system. The maximum range of the system is about 3 m, and the precision is 20% for range estimates and 30° for bearing estimates. The system also allows LoS communication with a nominal bandwidth of 40 kbps.¹ Finally, it also contains a number of other features, which are not relevant for the work presented here. For complete details of the foot-bot, as well as of the other robots developed in the Swarmanoid project the reader can refer to [2].

The foot-bots are derived from a previous robot called the *s-bot* [10]. Since the foot-bots are currently not yet available our work is based on *simulation*, whereby the simulator has been derived from an extensively tested s-bot simulator that has been refined and extended to include the new foot-bot features [3].

3 Use of LoS Routing Paths for Robot Navigation

The main idea in our approach is to use an ad hoc routing protocol to dynamically set up and maintain data routes in the LoS MANET between the event and the robot that can serve it. We assume that each event is represented by a robot that remains static at the event location and does all the communications for the event. The route information in the LoS MANET is then used directly or indirectly to guide robot navigation towards the event location.

3.1 AntHocNet, the MANET Routing Algorithm

To establish routes in the MANET, we make use of *AntHocNet*, a MANET routing algorithm based on ideas from *Ant Colony Optimization (ACO)* [5]. Here we give an high level overview of the algorithm. For more details, see [6].

In AntHocNet, a node s requiring a route to a destination d sends out a *reactive forward ant*. This is a control packet that has as a task to find a path to d . At any node i in the network, the reactive forward ant can be locally broadcast or unicast, depending on whether or not i has routing information available for d .

¹ These are the performance values of the implementation of the Ir-RB system at the time this paper has been written. Efforts to create an improved design of the Ir-RB system are taking place, and performance values are expected to be better in future versions.

When a node receives multiple copies of the same ant, it forwards only the first one it receives, and discards all subsequent copies. Once the ant reaches d , it is returned to its source node s , following the same path it came over. On its way back, the ant measures the quality of the path and sets up routing information towards the destination. Path quality is measured based on the signal strength of the wireless links along the path. Routing information takes the form of next hop pointers associated to relative goodness values based on the path quality measurements. These goodness values are called *pheromone values*, in accordance to the ACO inspiration of the algorithm, and are stored in routing tables.

Once the route is set up, the source node s starts a *route maintenance and improvement process*, in order to continuously adapt the existing route to the changes in the dynamic network and find new and better routes when possible. This process is based on two subprocesses: pheromone diffusion and proactive ant sampling. The aim of *pheromone diffusion* is to spread out pheromone information that was placed by the ants. Nodes periodically broadcast messages containing the best pheromone information they have available. Neighboring nodes receiving these messages can then derive new pheromone for themselves (using *information bootstrapping*, similar to Bellman-Ford updating schemes), and further forward it in their own periodic broadcasts. This way, a field of diffused pheromone is set up that points out possible routes towards the destination. However, since this information is based on the use of periodic (low-frequency) broadcast messages, it can temporarily contain erroneous information. This is where the second process, *proactive ant sampling*, comes in. At constant intervals, node s sends out *proactive forward ants*. Like reactive forward ants, these are control packets that try to find a route towards the destination. They follow the pheromone spread through the diffusion process. When they reach the destination, they travel back to the source setting up a route indicated by pheromone. This way, they update and validate the information found through the pheromone diffusion process and find new routes in the changing MANET.

In case of a route failure, AntHocNet sends link failure notification messages. When a node i perceives that a link has failed on an existing route, it broadcasts to its neighbors a message indicating the destination it has lost the route to. A neighbor receiving this message updates its routing information accordingly. If it observes the loss of a route due to this update, it sends out its own notification.

3.2 Network Routing and Robot Navigation

When a robot wants to serve a particular event, it uses the AntHocNet routing algorithm to set up a data route to the robot signaling the event. Once a route is set up, it amounts to a set of nodes (robots) connecting the service robot to the event in the LoS MANET, together with the (noisy) information of their relative (*distance, angle*) location. We foresee two possible ways of using this LoS routing information. The first is that the robot physically follows the data route in the network robot by robot. The other is that the full route information serves to calculate an estimate of the relative location of the event, and the robot moves directly in that direction.

Locally Following the Routing Path. In this approach, once the service robot has established a data route in the LoS MANET, it starts moving towards the estimated location of the nearest robot (next hop) in the data route. While moving, the robot continuously tries to re-sample the route in order to get a more up-to-date estimate of the route towards the event, and, more in particular, of the location of the next hop, which it is moving to. If the route is lost at any time (e.g. due loss of connectivity), the robot remains static and repeatedly tries to establish a new route. According to this behavior, all robots get continuous measurements of the relative distance and angle to each of their neighbors in the LoS network. Since these measurements are affected by precision errors, each robot aggregates them using moving averages:

$$\begin{aligned}\hat{d}_i^j(t) &= \gamma \hat{d}_i^j(t-1) + (1 - \gamma) d_i^j(t) \\ \hat{\alpha}_i^j(t) &= \gamma \hat{\alpha}_i^j(t-1) + (1 - \gamma) \alpha_i^j(t),\end{aligned}\quad (1)$$

where $\hat{d}_i^j(t)$ is robot i 's estimate at time t for the distance to neighbor robot j , and $\hat{\alpha}_i^j(t)$ is i 's estimate of the angle towards j with respect to its own orientation. $d_i^j(t)$ is the new measurement for the distance received by i at time t , and $\alpha_i^j(t)$ is the new measurement for the angle. $\gamma \in [0, 1[$ defines how quickly the local estimate is adapted to new measurements (we use $\gamma = 0.7$ in the experiments).

Having the robot physically follow the data route has a number of advantages and disadvantages. A first advantage is that it is a simple process. A second advantage is that it provides obstacle-free paths. This is because routes are composed of LoS links, that is, of a feasible path to the event. A disadvantage is that the robot can have difficulties following the path when it changes often and abruptly. This can happen when the robots move a lot or in the presence of obstacles. Another disadvantage is that the robot does not know where to move when there is no route available. This leads to low performance in cases of intermittent network connectivity, e.g. when there are few robots around or when obstacles block signals. A final disadvantage is that the path followed by the robot can be substantially longer than the shortest path, especially when the shortest path in the LoS MANET does not correspond to the geographic shortest path (e.g., this can easily happen when robot density is low [13]).

Following Path-Level Estimates of Destination Location. In this approach the constructed data route is used to give the searching robot an estimate of the relative distance and angle to the event location, so that it can move there directly without following the route. According to AntHocNet's behavior, to refresh the data route, the service robot periodically sends proactive forward ants towards the destination node (the event robot), which are then sent back to set up a new data route. On their way back, we let these ants gather the locally maintained estimates of the distance $\hat{d}_i^j(t)$ and angle $\hat{\alpha}_i^j(t)$ to each next hop and previous hop (see Equation 1) and combine them geometrically to make an estimation of the relative distance $D_i^n(t)$ and angle $A_i^n(t)$ to the event location n . We represent the path followed by the ant as $P = (1, 2, \dots, n-1, n)$, whereby node 1 is the searching robot and node n is the event location (so the

ant travels from n to 1). At any node $i < n$ on this path, $D_i^n(t)$ and $A_i^n(t)$ are incrementally calculated as follows, where the common index t is dropped for notational clarity:

$$D_i^n = \begin{cases} \hat{d}_i^n & \text{if } i = n - 1, \\ \sqrt{(\hat{d}_i^{i+1})^2 + (D_{i+1}^n)^2 - 2\hat{d}_i^{i+1}D_{i+1}^n \cos(A_{i+1}^n - \hat{\alpha}_i^{i+1})} & \text{if } i < n - 1. \end{cases} \quad (2)$$

$$A_i^n = \begin{cases} \hat{\alpha}_i^n & \text{if } i = n - 1, \\ \arccos \left[\frac{(\hat{d}_i^{i+1})^2 + (D_i^n)^2 - (D_{i+1}^n)^2}{2\hat{d}_i^{i+1}D_i^n} \right] & \text{if } i < n - 1. \end{cases}$$

Once the searching robot has received a first estimate of the distance D_1^n and angle A_1^n towards the event location, it starts moving in straight line towards its goal. As the robot is going, the routing algorithm keeps sending proactive ants regularly, making new estimates available. Having a continuous stream of new estimates is important to overcome errors in previous estimates and to keep an updated view of the event location. Errors in the estimate stem from two main causes. First of all, the event location estimate is based on a composition of local distance and angle estimates along the links of the paths, each of which contains some error, and therefore the total estimate has an error that increases with the number of hops. Hence, at large distances, the event location estimate only offers a rough guideline for the robot's movements, while at smaller distances, the estimate becomes more accurate. The second source of errors is due to the robot's own movements. As the robot is going, it needs to adapt the location estimates according to its own rotations and displacements, using feedback from the local odometry. This causes the estimate to gradually become less reliable. Therefore, the periodic sending of proactive ants is needed to keep renewing it.

Using estimates of the relative event location has the advantage that the robot is not directly dependent on the LoS route itself or on its persistence for its movements. It is sufficient to get a new estimate from time to time to update the global estimate. Figure 1 shows the sample of a typical behavior in the error of the distance estimate in the basic experimental setting (see next section). The error shows large fluctuations at the beginning, that is at large distances, but also a generic decreasing trend and a rapid convergence to zero when approaching the event location. On the other hand, this approach cannot guarantee an obstacle free path, and can run into problems in presence of many obstacles scattered along the straight line between the searching robot and the event location.

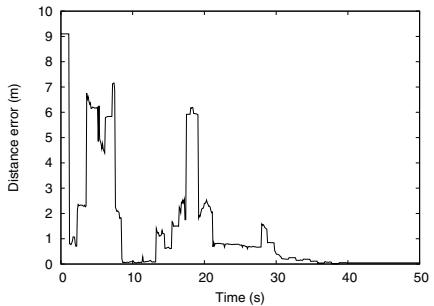


Fig. 1. Typical evolution of the error in the distance estimate for follow-estimate

4 Experimental Results

All tests are done using the Swarmanoid simulator [3]. For each test scenario we execute 30 independent runs. We report the average with 95% confidence interval of the time needed for the robot to reach the event and of the distance traveled compared to the straight line distance. We compare three different navigation behaviors: the two proposed ones (from now on referred to as *Follow route* and *Follow estimate*) and a *sweeping behavior*, used as a reference of the performance that is possible when no communication is used. In this behavior, the robot knows its location in the room at all times. It goes to the room corner that is closest to its start location and then starts scanning the room in straight lines parallel to one of the room walls, until it finds the destination. Moving steps in the direction of the other room walls are proportional to the radio range. We use an open space room of $10 \times 10 \text{ m}^2$. The service and the event robot are located in opposite corners. The other robots move according to the *random waypoint mobility model (RWP)* [9] in order to simulate the fact that they are involved in tasks of their own and their movements are independent from the task of guiding the searching robot. They choose a random destination, move to it, pause for some time and then choose a new random destination. This model fits well the movements of robots that service sequences of events. All robots are equipped with a minimal obstacle avoidance mechanism. We report experiments to study the effect of varying the number of robots and their speed, that equals to study the effect of varying network density and its topological changing rate. We also study the effect of changing the proactive ant send interval and that of increasing the fraction of packet losses during communications. Results concerning the effect of the presence of obstacles and of multiple events can be found in [7].

Effect of Scaling the Number of Robots. We investigate the influence of the number of robots on the ability of a searching robot to find an event location. We vary the number of robots from 10 up to 50. The speed of the robots is 0.15 m/s, the pause time of the RWP model is 6 s. The results are shown in figure 2. As can be seen from the graphs, a lower number of robots makes the task difficult for the communication based behaviors. This is because there is limited network

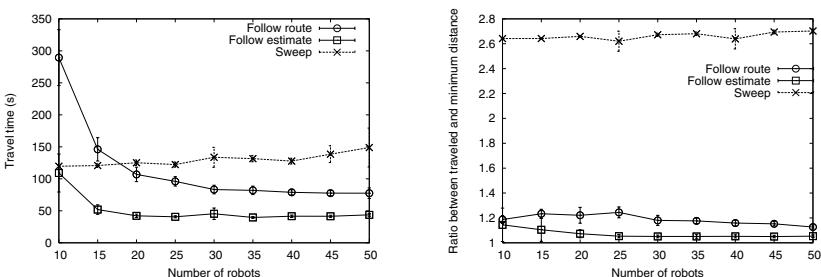


Fig. 2. Results for tests with increasing numbers of robots

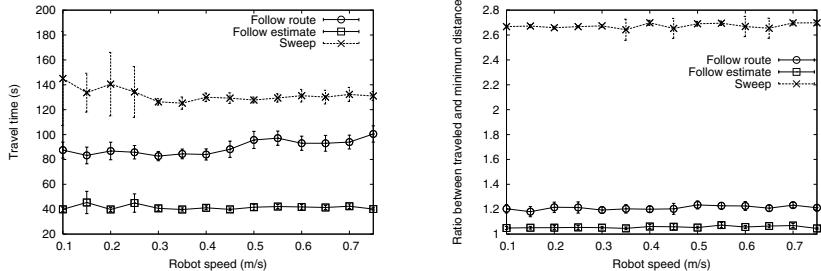


Fig. 3. Results for tests with increasing robot speed

connectivity and the task to establish and maintain a stable route is therefore difficult. This affects especially *Follow route*, which depends on the constant availability of a route: for this behavior, there is a strong increase in the travel time for the searching robot. For *Follow estimate*, the increase of the travel time is only visible for the lowest number of robots. Interestingly, the travel distance is not much affected for either of the behaviors. Overall, *Follow estimate* needs less time and produces shorter paths than *Follow route*. The *Sweep* behavior needs much more time and produces much longer travel distances, especially as the number of robots increases. This indicates the general usefulness of using telecommunications for navigation when a large group of robots is available.

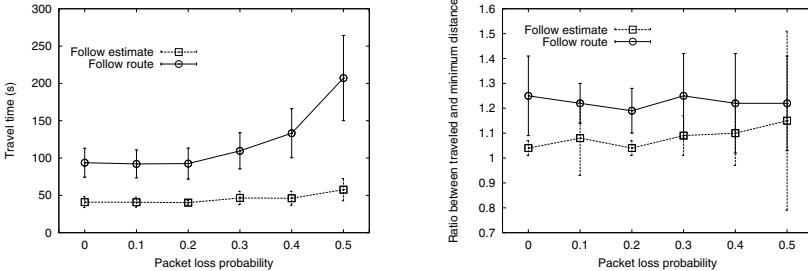
Effect of Robot Speed. The speed of the searching robot is set to 0.35 m/s. For the other robots, the speed is varied from 0.1 m/s up to 0.75 m/s. The total number of robots is fixed to 30. The results for increasing movement speed are shown in Figure 3. It is interesting to see that the speed of the robots has little influence on the performance. For *Follow route* there is a small increase in the robot travel time, but no noticeable effect on the traveled distance. For *Follow estimate*, there is no noticeable effect for either measure. The higher vulnerability of the *Follow route* behavior is to be expected, as high node mobility leads to higher variability and more frequent disconnections of the LoS MANET route, so that it becomes difficult to follow it hop by hop. The robustness of our approach with respect to robot speed is an important advantage for its deployment.

Effect of the Ant Send Interval. We evaluate the effect of changing the time interval in the periodic transmission of proactive forward ants. This parameter defines the frequency of route updates and hence also the load on the MANET. In previous tests, we used intervals of 1 s, here we run tests up to 10 s. The results are shown in Table 1. *Follow route* is unaffected by the ant send interval, while *Follow estimate* shows a slight decrease of performance with increasing intervals. This is because *Follow estimate* needs a continuous stream of ants to keep reducing the error on its estimate of the event location, while *Follow route* relies only on the presence of the route and can therefore more easily function with a lower frequency of proactive ants.

Table 1. Effect of changing the send interval of proactive ants

Proactive ant send interval	1	2	3	4	5	6	7	8	9	10
Follow route: Distance ratio	1.24	1.21	1.20	1.21	1.22	1.20	1.12	1.20	1.21	1.22
Follow route: Travel time	96	96	95	94	93	93	90	90	95	93
Follow estimate: Distance ratio	1.05	1.06	1.09	1.10	1.10	1.10	1.10	1.14	1.15	1.14
Follow estimate: Travel time	41	41	42	42	43	43	43	44	45	45

Effect of the Packet Losses. In the Swarmanoid simulator, the MAC and PHY layers are simulated with a probabilistic model based on two parameters defining the probabilities of packet loss, θ , and of signal interference (both set to 0 in the previous experiments). Here, we increase θ from 0 up to the extreme value of 0.5. An increase in θ results in the decrease of the ability of AntHocNet to gather up-to-date routing information. The results for the case of 30 robots are shown in Figure 4. *Follow estimate* shows no noticeable effect in travel time. In terms of distance, the average value is stationary but the variance gets larger and larger. In fact, in presence of large packet losses, the robot can be forced to rely on poor estimates for relatively long time periods, especially at the beginning of the search, when paths are longer. Therefore, the initial fluctuations shown in Figure 1 may become more severe in some scenarios. *Follow route* depends critically on the continual availability of route updates. Accordingly, its time performance shows a rapid degradation for $\theta \geq 0.3$. The distance performance is not much affected since the robot stops in absence of new route information.

**Fig. 4.** Results for tests with increasing probability of packet losses

5 Conclusions and Future Work

We have investigated the use of LoS wireless networking to support cooperative navigation in a swarm of mobile robots. We have based our work on the *foot-bots*, small robots developed in the Swarmanoid project [1], that are equipped with a LoS infrared device that can be used to both transmit data and return the relative distance and angle between two robots. We have proposed two solutions based on the use of a bio-inspired ad hoc network routing protocol to discover and maintain data routes between a robot and an event location in the LoS network. The established route is used for robot navigation in two different ways: in the first, the robot physically follows the route formed via the wireless connections,

in the other the route is used to make estimates of the relative position of the event. We ran a number of simulation experiments varying the number of robots and their speed, and studying the sensitivity of the algorithms to variations in the number of control packets used to gather routing information and in the probability of packet losses. Both algorithms showed an overall robust behavior, with the approach based on location estimates outperforming the other. In the future, we will consider also the other types of robots forming the 3D Swarmanoid and integrate the navigation function in a task allocation architecture.

References

1. Swarmanoid: Towards humanoid robotic swarms, <http://www.swarmanoid.org>
2. Hardware design. Internal Deliverable D4 of IST-FET Project Swarmanoid funded by the European Commission under the FP6 Framework (2007)
3. Simulator prototype. Internal Deliverable D7 of IST-FET Project Swarmanoid funded by the European Commission under the FP6 Framework (2008)
4. Di Caro, G.A., Ducatelle, F., Gambardella, L.M.: A simulation study of routing performance in realistic urban scenarios for manets. In: Proc. of the 6th Int. Workshop on Ant Algorithms and Swarm Intelligence. Springer, Heidelberg (2008)
5. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for distributed discrete optimization. *Artificial Life* 5(2), 137–172 (1999)
6. Ducatelle, F., Di Caro, G., Gambardella, L.M.: Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications (IJCIA)* 5(2) (2005)
7. Ducatelle, F., Di Caro, G.A., Gambardella, L.M.: Robot navigation in a networked swarm. In: Xiong, C., Liu, H., Huang, Y., Xiong, Y. (eds.) ICIRA 2008. LNCS (LNAI), vol. 5314. Springer, Heidelberg (2008)
8. Farinelli, A., Iocchi, L., Nardi, D.: Multirobot systems: a classification focused on coordination. *IEEE Trans. on Sys., Man, and Cyb., Part B* 34(5) (2004)
9. Johnson, D.B., Maltz, D.A.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: Mobile Computing. Kluwer, Dordrecht (1996)
10. Mondada, F., Pettinari, G., Guignard, A., Kwee, I., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L.M., Dorigo, M.: SWARM-BOT: a new distributed robotic concept. *Autonomous Robots* 17(2–3) (2004)
11. Nouyan, S., Dorigo, M.: Chain based path formation in swarms of robots. In: Proc. of the 5th Int. Workshop on Ant Algorithms and Swarm Intelligence (2006)
12. Payton, D., Daily, M., Estowski, R., Howard, M., Lee, C.: Pheromone robotics. *Autonomous Robots* 11(3) (November 2001)
13. Sit, T.C.H., Liu, Z., Ang Jr., M.H., Seah, W.K.G.: Multi-robot mobility enhanced hop-count based localization in ad hoc networks. *Robotics and Autonomous Systems* 55(3), 244–252 (2007)

An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks

Fatma Corut Ergin, Aysegül Yayımlı, and Şima Uyar

Istanbul Technical University

fcorut@eng.marmara.edu.tr,

{gencata, etaner}@itu.edu.tr

Abstract. The high capacity of fibers used in optical networks, can be divided into many channels, using the WDM technology. Any damage to a fiber causes all the channels routed through this link to be broken, which may result in a serious amount of data loss. As a solution to this problem, the virtual layer can be mapped onto the physical topology, such that, a failure on any physical link does not disconnect the virtual topology. This is known as the survivable virtual topology mapping problem. In this study, our aim is to design an efficient evolutionary algorithm to find a survivable mapping of a given virtual topology while minimizing the resource usage. We develop and experiment with different evolutionary algorithm components. As a result, we propose a suitable evolutionary algorithm and show that it can be successfully used for this problem. Overall, the results are promising and promote further study.

Keywords: Optical networks, WDM, survivable virtual topology design, evolutionary algorithms, constraint optimization.

1 Introduction

Today, optical networking [1] is the most effective technology to meet the high bandwidth network demand. The high capacity of fiber used in optical networks, can be divided into hundreds of different transmission channels, using the WDM (wavelength division multiplexing) technology. Each of these channels work on different wavelengths and can be associated with a different optical connection. The upper layers (IP, Ethernet, etc.) can transmit data using these optical connections. This architecture is known as IP-over-WDM, or Ethernet-over-WDM.

End-to-end optical connections used by the packet layer (IP, Ethernet, etc.) are called lightpaths. Since the fibers on the physical topology allow traffic flow on different wavelengths, more than one lightpath, each operating on different wavelengths, can be routed on a single fiber. All the lightpaths set up on the network form the virtual topology (VT). Given the physical parameters of the network (physical topology, optical transceivers on the nodes, wavelength numbers on the fibers, etc.) and the mean traffic rates between nodes, the problem of designing the lightpaths to be set up on the physical topology is known as

the VT design problem. VT mapping problem, which is a subproblem of VT design, is to find a proper route for each lightpath of the given VT and to assign wavelengths to these lightpaths.

Any damage to a physical link (fiber) on the network causes all the lightpaths routed through this link to be broken. Since huge amount of data (40 Gb/s) can be transmitted over each of these lightpaths, a fiber damage may result in a serious amount of data loss. Two different approaches can be used to avoid data loss: 1. Survivable design of the physical layer, 2. Survivable design of the virtual layer [1]. The first approach is the problem of designing a backup link/path for each link/path of the virtual layer. The second approach is the problem of designing the virtual layer such that the virtual layer remains connected in the event of a single or multiple link failure. While the first approach provides faster recovery for time-critical applications (such as, IP phone, telemedicine) by reserving more resources; the second approach, i.e. the survivable VT design, which has attracted a lot of attention in recent years, aims to protect data communication using less resources. In this study, our main aim is to design an efficient evolutionary algorithm (EA) to find a survivable mapping of a given VT while minimizing the resource usage.

To illustrate VT mapping problem, assume that we have a physical network topology as in Figure 1.a and a virtual network topology to be routed on this physical topology as in Figure 1.b. If we route this VT as in Figure 1.c we obtain a survivable mapping, that is, a failure on any physical link does not disconnect the VT. However, if the routing of only one lightpath is changed, e.g., as in Figure 1.d, we end up with an unsurvivable mapping. In this case, if a failure occurs on the physical link between nodes 4 and 5, the nodes connected with lightpaths *b* and *g* will not be able to find an alternative path to communicate.

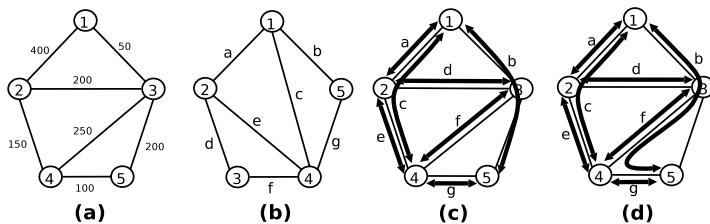


Fig. 1. a. Physical Topology, b. Virtual Topology, c. Survivable Mapping, d. Unsurvivable Mapping

The VT mapping problem is known to be NP-complete [2]. Because of its complexity, for real-life sized networks, it is not possible to solve the problem optimally in an acceptable amount of time using classical optimization techniques. Therefore, heuristic approaches should be used. In this study, we chose EAs due to their successful applications on NP-complete problems. We proposed new mutation operators and fitness evaluation methods and compared their performance to design an effective EA specific to this problem.

The structure of the search space [3] plays an important role on the performance and design of EAs. In order to support our effective EA design, we examined the search space structure of this problem through a correlation length analysis of our different mutation operators and fitness evaluation methods. We performed a comprehensive experimental study for different physical and virtual network topologies. As a result of these experiments, we show that EAs can be successfully used in solving VT mapping problem, and we recommend an appropriate selection of EA components (solution representation, mutation operator, and fitness evaluation method).

2 Problem Definition and Related Literature

In this work, given the physical and the virtual network topologies, our aim is to find a survivable mapping of the VT. Survivable mapping of VT is to find a route for each lightpath, such that in case of a single physical link failure the VT remains connected.

2.1 Formal Problem Definition

The physical topology is composed of a set of nodes $N = \{1..N\}$ and a set of edges E , where (i, j) is in E if there is a link between nodes i and j . Each link has a capacity of W wavelengths. The VT, on the other hand, has a set of virtual nodes N_L , which is a subset of N , and virtual edges (lightpaths) E_L , where an edge (s, t) exists in E_L if both node s and node t are in N_L and there is a lightpath between them.

An ILP formulation of survivable lightpath routing of a VT on top of a given physical topology is given in [2]. Based on this formulation, a number of different objective functions can be considered for the problem of survivable mapping. The simplest objective is to minimize the number of physical links used. Another objective is to minimize the total number of wavelength-links used in the whole physical topology. A wavelength-link is defined as a wavelength used on a physical link. Our choice as the objective is the latter one, since it gives a better idea of the actual resource usage.

The aim of lightpath routing is to find a set of physical links that connect the nodes of the lightpaths. Our objective is to minimize the total number of wavelength-links used in the whole physical topology, subject to two constraints.

- a)Survivability constraint: The survivability constraint states that for all proper cuts of the VT, the number of cut-set links flowing on any given physical link is less than the size of the cut-set. This means that all the lightpaths of a cut-set cannot be routed using the same physical link.
- b)Capacity constraints: This constraint ensures that the number of wavelengths on a physical link does not exceed its capacity W .

2.2 Related Literature

The survivable VT mapping problem was first addressed as Design Protection [4] in the literature. In this first study, tabu search was used to find the minimum number of source-destination pairs that become disconnected in the event of a physical link failure. Nucci et. al. also used tabu search to solve the survivable VT design problem [5]. The constraints in this study include transmitter and receiver constraints as well as wavelength capacity constraints. In other heuristic approaches to VT mapping Ducatelle et. al. [6] studied the problem using a local search algorithm, while Kurant and Thiran [7] used an algorithm that divides the survivable mapping problem into subproblems. There are a few studies on VT mapping [8] and design [9] using EA, however, the survivability is not considered in any of them.

Modiano and Narula-Tam used ILP (Integer Linear Programming) to solve the VT mapping problem [2]. They added the survivability constraint in the problem formulation, such that, no physical link is shared by all virtual links belonging to a cut-set of the VT graph. However, they did not consider the capacity constraint. Their objective was to minimize the number of wavelengths used. For the cases when ILP cannot find an optimum solution in a reasonable amount of time due to the problem size, Modiano et. al. proposed two relaxations to ILP, which consider only small-sized cut-sets. These relaxations reduce the problem size; however, they may lead to suboptimal solutions. In order to overcome the long execution time problem in ILP formulation, Todimala and Ramamurthy proposed a new ILP formulation and they solved the problem for networks of up to 24 nodes [10]. In [10], besides the physical network and the virtual network topologies, the shared risk link groups should be known in advance.

3 Proposed Evolutionary Algorithm

Designing a solution representation that is well-suited to the problem is crucial in EA performance. VT problem can be seen as a search for the best routing of lightpaths through physical links. Therefore, we use a solution encoding inspired from [8]. For this encoding, first, the k -shortest paths for each lightpath are determined. Then, a solution candidate is represented as an integer string of length l , where l is the number of lightpaths in the VT. Each gene gives the index of the shortest path for the corresponding lightpath. Genes can take on values between $[1..k]$ where k is the predefined number of shortest paths for the lightpaths.

A steady-state EA with duplicate elimination is used where a new individual is generated and inserted into the population at each iteration. After a random initial population generation, the EA operators are applied to the solutions until a predefined number of fitness evaluations are executed. Mating pairs are selected through binary tournament selection. The two selected individuals undergo reproduction. Reproduction consists of uniform crossover and problem specific mutation operators. The offspring replaces the worst individual in the population, if its fitness is better.

3.1 Mutation Operators

We define two different mutation operators. The first one is a simple random-reset mutation, called gene mutation. In this type of mutation, the value of a gene is randomly reset to another value within the allowed range. The second is a problem-specific mutation operator, called path mutation. It considers the physical link similarities between the shortest paths of each lightpath. If mutation occurs on a gene, its current value is replaced by the index of the least similar shortest path for the corresponding lightpath, similarity being defined as the number of common physical links. This mutation operator aims to preserve diversity in the population.

3.2 Fitness Evaluation Methods

Our objective is to minimize the total cost of resources used throughout the network. This cost is evaluated in two different ways: (1) by considering the actual lengths of the physical links (link cost), and (2) by counting the number of physical links used (hop count).

The constraints for the problem, i.e. the survivability and the capacity constraints, are explained in section 2.1. Violations of these constraints are included as penalties to the fitness function. We define three different fitness functions based on three different survivability constraint violation handling approaches.

In order to determine if the solution is survivable or not, each physical link is deleted from the physical network one by one. If any of the lightpaths becomes disconnected in the event of a broken physical link, the solution is taken as unsurvivable. The penalty for an unsurvivable solution is determined in three different ways:

1. The total number of physical links, whose failure results in the network unsurvivability.
2. The sum of the total number of lightpaths that become disconnected in the event of each physical link failure [4,6].
3. The maximum of the total number of lightpaths that become disconnected in the event of each physical link failure [4].

Each of the fitness evaluation methods we define, f_1 , f_2 , and f_3 , use the penalty calculation methods above, respectively. In the first fitness evaluation method (f_1), the connectivity of the graph is checked for the failure of each physical link, which needs an algorithmic complexity of $O(e \cdot n^3)$. On the other hand, for the second and the third fitness evaluation methods (f_2 and f_3), a shortest path algorithm is applied for the failure of each physical link, which means $O(l \cdot (e+n) \cdot \log n)$ algorithmic complexity. Here, e is the number of physical links, n is the number of nodes, and l is the number of lightpaths.

A capacity constraint violation adds a penalty value which is proportional to the total number of physical links which are assigned more lightpaths than the predetermined wavelength capacity. This penalty is multiplied with a penalty factor and added to the fitness of the solution.

The following example illustrates the application of the mutation operators and the fitness evaluation techniques.

Consider the physical and virtual topologies given in Figure 1. The first 4 shortest paths calculated based on hop counts and based on link costs can be seen in Table 1. Here, the first column shows the lightpaths as source-destination node pairs. Four shortest paths found using hop counts are given in the first four columns, and 4 shortest paths found using link costs are given in the next four columns.

Table 1. Four different shortest paths for the lightpaths of the example virtual topology given in Figure 1

lightpath	hop count				link cost			
	sp ₁	sp ₂	sp ₃	sp ₄	sp ₁	sp ₂	sp ₃	sp ₄
1-2 (a)	1-2	1-3-2	1-3-4-2	1-3-5-4-2	1-3-2	1-2	1-3-4-2	1-3-5-4-2
1-4 (b)	1-2-4	1-3-4	1-3-2-4	1-3-5-4	1-3-4	1-3-5-4	1-3-2-4	1-2-4
1-5 (c)	1-3-5	1-2-4-5	1-2-3-5	1-3-4-5	1-3-5	1-3-4-5	1-3-2-4-5	1-2-4-5
2-3 (d)	2-3	2-1-3	2-4-3	2-4-5-3	2-3	2-4-3	2-1-3	2-4-5-3
2-4 (e)	2-4	2-3-4	2-1-3-4	2-3-5-4	2-4	2-3-4	2-3-5-4	2-1-3-4
3-4 (f)	3-4	3-2-4	3-5-4	3-1-2-4	3-4	3-5-4	3-2-4	3-1-2-4
4-5 (g)	4-5	4-3-5	4-2-3-5	4-2-1-3-5	4-5	4-3-5	4-2-3-5	4-2-1-3-5

Assume we have an individual encoded as [1 1 2 3 1 1 2]. This encoding means that the first lightpath uses the 1st shortest path (1-2), the second one uses the 1st shortest path (1-2-4), and the third one uses 2nd shortest path (1-2-4-5), etc. If we sum up the number of wavelength-links used in this solution, we have a total of 12 wavelength-links for hop count evaluation, and 14 wavelength-links for link cost evaluation. For example, in f_1 , the number of physical links which leave the network disconnected in the event of a failure are counted. For this sample individual, if a failure occurs on the physical links connecting nodes 1-2, 2-4, and 3-4, the virtual topology becomes disconnected. Thus, a penalty for 3 survivability violations ($p * 100$) is added to the fitness, where p is the penalty factor taken as 100 in this example. Six different fitness values calculated using three different evaluation functions and two different cost metrics are given in Table 2. When calculating the fitness, the link costs given in Figure 1 are normalized dividing by 100.

Table 2. Fitness values for individual [1 1 2 3 1 1 2] calculated using three different evaluation functions and two different cost metrics. penalty factor=100.

	hop count	link cost
f_1	$12+100*3=312$	$22.5+100*2=214$
f_2	$12+100*9=912$	$22.5+100*5=514$
f_3	$12+100*4=412$	$22.5+100*3=314$

Assume a path mutation occurs on the second gene of this sample individual, the new individual becomes [1 2 2 3 1 1 2] or [1 4 2 3 1 1 2] with equal probability, if shortest paths are calculated according to hop count. However, if link costs are used instead, the individual becomes [1 4 2 3 1 1 2].

4 Experimental Design

To determine a good operator and evaluation method combination, we performed a series of experiments on our newly designed mutation operators and fitness evaluation methods. In these experiments, we used three metrics for performance comparisons, namely success rate (sr), first hit time (fht), and correlation length. Success rate is defined as the percentage of program runs in which a survivable mapping that does not violate the capacity constraint is found. First hit time is the first iteration during which the best solution is encountered.

Ruggedness [3] is commonly used to analyze the structure of fitness landscapes and algorithm behavior. The autocorrelation function (ACF) is one of the simplest and the most commonly used techniques for analyzing the ruggedness of a landscape. The ACF looks at the amount of fitness correlation between the points in the search space and takes on values between $[-1, 1]$. Values close to 1 denote a high positive correlation and values close to 0 show low correlation. The ACF value ρ_s is calculated as

$$\rho_s = \frac{\sum_{t=1}^{T-s} (f_t - \bar{f})(f_{t+s} - \bar{f})}{\sum_{t=1}^T (f_t - \bar{f})^2} , \quad \lambda = -\frac{1}{\ln(|\rho_s|)} \quad (1)$$

where, s is the step size, T is the total number of sample points, f_t is the fitness of the t . solution, and \bar{f} is the average fitness of all the points.

To compare the ruggedness of different landscapes, usually the correlation length, λ , as defined in Eq. 1 is used. A high correlation length means a smoother landscape, while a low correlation length shows a more rugged landscape.

4.1 Experimental Setup

For the experiments, we used two different physical topologies: the 14-node 24-link NSF network and a 24-node 43-link network (see [1] chapter 11 pp.557). For each physical topology we created 10 random VTs with average connectivity degrees of 3, 4, and 5. We assumed 10 wavelengths per physical link.

In the EA performance tests, we considered a maximum fitness evaluation count of 5000, mutation probability of $1/l$, where l is the number of lightpaths, crossover probability of 1.0, and population size of 100, and we ran the program 20 times for each parameter set.¹ In the landscape analysis tests, we used 1000 random walks and 50 runs. A penalty factor of 200 is used in the tests using hop count for shortest path calculation, and 300 in the tests using link cost.

4.2 Experimental Results

All EA component combinations we tested were able to solve the problem for NSF network for all 30 VTs, i.e., they were able to find survivable solutions of equal quality for all VTs. Since, the NSF network is a fairly simple and sparse graph, the results do not show meaningful differences between the tested

¹ On average a feasible solution is obtained for this problem in less than half a minute.

combinations. Therefore, in this paper we report the results for the 24-node 43-link network.

The results of the experiments are given in Tables 3, 4, and 5. Table 3 shows the success rates, Table 4 shows the correlation lengths, and Table 5 shows the first hit times. In all the tables, f_i denotes the corresponding results for the i^{th} fitness evaluation method. In the top half of the tables, the results obtained using the gene mutation are given, whereas in the bottom half, the path mutation results are given. Also, the results for three connectivity degrees, 3, 4, and 5, can be seen in the tables. For the correlation length and the EA first hit time results, we also showed the standard errors of the means (e) in the tables.

Table 3. Success rates for 24-node network

			5 shortest paths			10 shortest paths			15 shortest paths		
			hop count	link cost		hop count	link cost		hop count	link cost	
			f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
gene	3	89	62	19.5	71	57	18.5		89.5	95	51.5
	4	90	90	88	90	90	78.5		100	100	97
	5	100	100	100	100	100	100		100	100	100
path	3	89	62.5	29	73	59.5	23		93	95	61
	4	90	90	90	90	89.5	88.5		100	100	100
	5	100	100	100	100	99.5	99		100	100	100

From the tables, we can see that there is a difference in all the combinations for smaller shortest path counts and low connectivity degrees. These are relatively difficult problems because the probability of finding potential mappings increases with the node degrees and the number of alternative shortest paths.

As can be seen in Table 3 the performance of the third fitness evaluation method (f_3) is the worst of all. This is confirmed by Table 4, where f_3 has lower correlation lengths than f_1 and f_2 , showing a more rugged landscape.

Table 4. Average and standard error of correlation lengths

			5 shortest paths			10 shortest paths			15 shortest paths		
			hop count	link cost		hop count	link cost		hop count	link cost	
			f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
gene	3	λ	15.3	16.2	9.4	15.5	16.5	11.4	15.2	16.5	10.1
	4	λ	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.2	0.1
	5	λ	18.3	20.5	12.5	18.7	20.4	13.5	17.2	20.4	11.9
path	3	λ	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.1
	4	λ	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.1
	5	λ	17.0	22.8	13.9	18.7	23.8	14.8	16.4	22.8	13.3
	3	e	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	4	e	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	5	e	0.2	0.3	0.2	0.2	0.3	0.2	0.2	0.3	0.2

A difference between f_1 and f_2 can be seen for the 3 connected virtual topology tests. We can say f_1 performs better than the others on relatively difficult problems. In order to confirm this result, we created 100 different virtual topologies of connectivity degree 3 and ran the program 100 times for each of these

topologies, for 5 shortest paths, gene mutation, and hop count. We applied a 2 sample 2-tailed t-test with a significance level of 0.05 and saw a statistically significant difference between these two fitness evaluation methods. However, if the algorithmic complexity of fitness evaluation methods as explained in section 3.2 are considered, we can say that f_2 is better. Therefore, f_2 should be preferred for virtual topologies having larger degrees of connectivity.

A difference can be seen between hop count and link cost results for f_1 and f_2 in Table 3. However, as a result of the t-test as in the previous paragraph, we cannot say that there is a statistically significant difference between them. Similarly, there is no difference in their landscapes as given in Table 4. If we consider the first hit counts, we can prefer hop count to link cost.

A difference between gene mutation and path mutation can be seen in Table 3. However, again as a result of the same type of t-test as in the previous paragraphs, we cannot say that there is a statistically significant difference between them. However, if we look at Table 5, we can see that the first hit times of path mutation is lower than gene mutation. In Table 4, it can be seen that the correlation lengths for path mutation is less than the gene mutation. This is an expected result, since the neighborhood definition for path mutation means the most faraway results.

Table 5. Average and standard error of EA first hit times

		5 shortest paths				10 shortest paths				15 shortest paths				
		hop count		link cost		hop count		link cost		hop count		link cost		
		f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	
gene	3	fht	2735	2715	4012	3530	4176	4256	4815	4834	4670	4666	4901	4895
	4	fht	53.56	72.75	42.58	84.23	38.55	34.4	14.56	13.08	20.71	18.69	9.97	8.17
	5	fht	48.15	55	31.35	61.08	33.39	27.9	7.35	8.8	13.94	13.53	5.47	8.29
path	3	fht	3250	3382	4828	4573	4691	4764	4916	4921	4850	4871	4940	4923
	4	fht	47.1	50.04	14.04	53.59	19.32	14.77	5.76	6.56	9.1	8.62	5.24	16.92
	5	fht	56.44	71.42	51.19	74.09	46.68	45.81	31.44	28.04	39.27	33.44	22.85	16.04

		5 shortest paths				10 shortest paths				15 shortest paths				
		hop count		link cost		hop count		link cost		hop count		link cost		
		f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	
gene	3	fht	2722	2875	3653	3334	3912	3993	4499	4624	4322	4442	4750	4814
	4	fht	2621	2945	3964	3861	3849	4003	4731	4758	4469	4492	4867	4884
	5	fht	45.42	59.4	45.1	62.98	45.65	45.03	19.75	20.03	30.44	25.22	9.37	8.72
path	3	fht	2937	3146	4516	4367	4321	4415	4881	4878	4738	4768	4892	4915
	4	fht	48.31	49.07	29.35	53.73	34.82	31	10.35	10.76	16.15	15.61	7.87	6.71
	5	fht												

As a summary of the experiments, using path mutation, hop count, f_2 (f_1 in sparse virtual topologies) can be recommended as components for an effective EA for the survivable VT mapping problem.

5 Conclusion and Future Work

Our main aim in this study was to design an efficient evolutionary algorithm to find a survivable mapping of a given virtual topology while minimizing the resource usage. We experimented with different evolutionary algorithm components, and developed three fitness evaluation methods, two cost metrics and two mutation operators. We used a very simple search space structure analysis technique to support our results. As a result of experiments, we designed a

suitable evolutionary algorithm and showed that evolutionary algorithms can be successfully used for the survivable virtual topology mapping problem. Overall, the results are promising and promote further study to improve the EA performance. As future work, we plan to use better metrics for landscape analysis and explore more sophisticated nature inspired heuristics for this problem.

References

1. Mukherjee, B.: Optical WDM Networks. Springer, New York (2006)
2. Modiano, E., Narula-Tam, A.: Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks. *IEEE Journal on Selected Areas in Communications* 20(4), 800–809 (2002)
3. Reeves, C.R.: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. In: Burke, E.K., Kendall, G. (eds.) *Fitness Landscapes*, ch. 19. Springer, Heidelberg (2005)
4. Armitage, J., Crochat, O., Le Boudec, J.-Y.: Design of a Survivable WDM Photonic Network. In: INFOCOM 1997. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Driving the Information Revolution (1997)
5. Nucci, A., Sanso, B., Crainic, T., Leonardi, E., Marsan, M.A.: Design of Fault-Tolerant virtual Topologies in Wavelength-Routed Optical IP Networks. In: Proceedings of IEEE Globecom (2001)
6. Ducatelle, F., Gambardella, L.M.: A Scalable Algorithm for Survivable Routing in IP-Over-WDM Networks. In: First International Conference on Broadband Networks (BROADNETS 2004), (2004)
7. Kurant, M., Thiran, P.: Survivable Mapping Algorithm by Ring Trimming (SMART) for Large IP-over-WDM Networks. In: Proceedings of BroadNets (2004)
8. Banerjee, N., Sharan, S.: An EA for Solving the Single Objective Static RWA Problem in WDM Networks. In: ICISIP (2004)
9. Saha, M., Sengupta, I.: A GA Based Approach for Static Virtual Topology Design in Optical Networks, Indicon (2005)
10. Todimala, A., Ramamurthy, B.: A Scalable Approach for Survivable Virtual Topology Routing in Optical WDM Networks. *IEEE Journal on Selected Areas in Communications* 25(6) (August 2007)

Extremal Optimization as a Viable Means for Mapping in Grids

Ivanoe De Falco^{1,*}, Antonio Della Cioppa², Domenico Maisto¹,
Umberto Scafuri¹, and Ernesto Tarantino¹

¹ Institute of High Performance Computing and Networking,

National Research Council of Italy (ICAR-CNR)

Via P. Castellino 111, 80131 Naples, Italy

Tel.: +39-081-6139524; Fax: +39-081-6139531

{ivanoe.defalco,domenico.maisto,

umberto.scafuri,ernesto.tarantino}@na.icar.cnr.it

² Natural Computation Lab, DIIIE, University of Salerno,

Via Ponte don Melillo 1, 84084 Fisciano (SA), Italy

adellacioppa@unisa.it

Abstract. An innovative strategy, based on Extremal Optimization, to map the tasks making up a user application in grid environments is proposed. Differently from other evolutionary-based methods which simply search for one site onto which deploy the application, our method deals with a multisite approach. Moreover, we consider the nodes composing the sites as the lowest computational units and we take into account their actual loads. The proposed approach is tested on a group of different simulations representing a set of typical real-time situations.

1 Introduction

A computational grid is a decentralized heterogeneous multisite system which aggregates multi-owner resources spread across multiple domains and provides a platform for exploiting various computational resources over wide area networks. In fact, this system enables the virtualization of distributed computing, so as to create a single powerful collaborative problem-solving environment. One of the concerns in implementing a computational grid environment is how to effectively map tasks of a parallel application onto resources in order to gain increased utilization in this highly heterogeneous environment. Thus, given a grid constituted by several sites, each of which contains a specified number of nodes, the communicating tasks could be conveniently assigned to the grid nodes which, on the basis of their characteristics and load conditions, turn out to be more suitable to execute them. However, maximizing the global performance remains an open challenge when the tasks of a parallel application are executed on non-dedicated grid systems in concurrence with the local workload. The problem, already complex for the high number of tasks and resources involved, becomes even more challenging if, among the suitable resources, those able to guarantee the co-scheduling must be selected to avoid possible deadlock conditions [1].

* Corresponding author.

In many cases the search for just one single site onto which to allocate all the application tasks could be insufficient for fulfilling all these needs. Thus, a multisite mapping tool, able to choose among resources spread over multiple sites and to match applications demands with the networked grid computing resources, must be designed.

Since mapping is an NP-complete problem [2], several evolutionary-based techniques have been used to face it in heterogeneous or grid environments [3, 4, 5, 6]. It should be noted that the problem shows linear correlations among variables due to both communications and multitasking executions. Moreover, to find a solution, it is possible to assume that the objective function consists of single contributions, one for each variable, so that the use of a coevolutionary approach based on a parameter-wise evaluation is feasible. Starting from the above considerations and differently from classical approaches, we propose an innovative method based on Extremal Optimization (EO) [7], a coevolutionary algorithm successfully applied to NP-hard combinatorial problems [8].

Unlike the other existing evolutionary techniques which simply search for just one site onto which map the application, our method deals with a multisite approach. Moreover, as a further distinctive issue with respect to other approaches [9], we consider the site nodes as the lowest computational units and we take into account their actual loads. The final goal consists in the design of a software system capable of performing a mapping so as to minimize, in terms of time, the use of the grid resource it has to exploit at the most as well as to complete the tasks of a given application in a minimum amount of time.

In the following, section 2 outlines EO method and section 3 better describes our view on the mapping problem and its formalization in terms of EO. Section 4 reports on the test problems experienced and shows the findings achieved, while section 5 contains our conclusions.

2 Extremal Optimization

In nature, highly specialized, complex structures often emerge when their most inefficient elements are selectively driven to extinction. Such a view is based on the principle that evolution progresses by selecting against the few most poorly adapted species, rather than by expressly breeding those species well adapted to the environment. This idea has been applied successfully in the Bak–Sneppen model [10] which shows the emergence of Self–Organized Criticality (SOC) in ecosystems. According to that model, each component of an ecosystem corresponds to a species which is characterized by a fitness value. The evolution is driven by a process where the least fit species together with its closest dependent species are selected for adaptive changes. As the fitness of one species changes, those of its neighbors are affected. Thus, species coevolve and the resulting dynamics of this extremal process exhibits the characteristics of SOC, such as *punctuated equilibrium* [10].

Extremal Optimization was proposed by Boettcher and Percus and draws upon the Bak–Sneppen mechanism, yielding a dynamic optimization procedure free of selection parameters. It represents a successful method for the study

of NP-hard combinatorial and physical optimization problems [7, 8] and a competitive alternative to other nature-inspired paradigms such as Simulated Annealing, Evolutionary Algorithms, Swarm Intelligence and so on, typically used for finding high-quality solutions to such NP-hard problems. Differently from the well-known paradigm of Evolutionary Computation (EC), which assigns a given fitness value to the whole set of the components of a solution based upon their collective evaluation against a cost function and operates with a population of candidate solutions, EO works with one single solution S made of a given number of components s_i , each of which is a variable of the problem and is thought to be a species of the ecosystem. Once a suitable representation is chosen, by assuming a predetermined interaction among these variables, a fitness value ϕ_i is assigned to each of them. Then, at each time step the overall fitness Φ of S is computed and this latter is evolved, by randomly updating only the worst variable, to a solution S' belonging to its neighborhood $\text{Neigh}(S)$. This last is the set of all the solutions that can be generated by randomly changing only one variable of S by means of a uniform mutation. However, EO is competitive with respect to other EC techniques if it can randomly choose among many $S' \in \text{Neigh}(S)$. When this is not the case, EO leads to a deterministic process, i.e., gets stuck in a local optimum. To avoid this behavior, Boettcher and Percus introduced a probabilistic version of EO based on a parameter τ , i.e., τ -EO. According to it, for a minimization problem, the species are firstly ranked in increasing order of fitness values, i.e., a permutation π of the variable labels i is found such that: $\phi_{\pi(1)} \leq \phi_{\pi(2)} \leq \dots \leq \phi_{\pi(n)}$, where n is the number of species. The worst species s_j is of rank 1, i.e., $j = \pi(1)$, while the best one is of rank n . Then, a distribution probability over the ranks k is considered as follows:

$$p_k \propto k^{-\tau}, \quad 1 \leq k \leq n$$

for a given value of the parameter τ . Finally, at each update a generic rank k is selected according to p_k so that the species s_i with $i = \pi(k)$ randomly changes its state and the solution moves to a neighboring one $S' \in \text{Neigh}(S)$ unconditionally. Note that only a small number of variables change their fitness, so that only a few connected variables need to be re-evaluated and re-ranked. The only parameters are the maximum number of iterations N_{iter} and the probabilistic selection value τ . For minimization problems τ -EO proceeds as in the Algorithm 1.

3 Mapping in Grids

To focus mapping we need information on the number and on the status of both accessible and demanded resources. We assume to have an application divided into P tasks to be mapped on n nodes. Each node is identified by an integer value in the range $[0, N - 1]$, where N is the total number of available grid nodes.

We need to know *a priori* the number of instructions α_i computed per time unit on each node i , and the communication bandwidth β_{ij} between any couple of nodes i and j . This information is supposed to be contained in tables based either on statistical estimations in a particular time span or gathered by

Algorithm 1. Pseudocode of the τ -EO algorithm

```

begin
    initialize configuration  $S$  at will
    set  $S_{best} := S$ 
    while maximum number of iterations  $N_{iter}$  not reached do
        evaluate  $\phi_i$  for each variable  $s_i$  of the current solution  $S$ 
        rank the variables  $s_i$  based on their fitness  $\phi_i$ 
        choose the rank  $k$  according to  $k^{-\tau}$  so that the variable  $s_j$  with  $j = \pi(k)$  is selected
        choose  $S' \in \text{Neigh}(S)$  such that  $s_j$  must change
        accept  $S := S'$  unconditionally
        if  $\Phi(S) < \Phi(S_{best})$  then
            set  $S_{best} := S$ 
        end if
    end while
    return  $S_{best}$  and  $\Phi(S_{best})$ 
end

```

tracking periodically and by forecasting dynamically resource conditions [11, 12]. For example, in the Globus Toolkit [13], a standard grid middleware, similar information is gathered by the Grid Index Information Service (GIIS) [12].

Since grids address non dedicated-resources, their own local workloads must be considered to evaluate the computation time of the tasks. There exist several prediction methods to face the challenge of non-dedicated resources [14, 15]. We suppose to know the average load $\ell_i(\Delta t)$ of the node i at a given time span Δt with $\ell_i(\Delta t) \in [0.0, 1.0]$, where 0.0 means a node completely discharged and 1.0 a node locally loaded at 100%. Hence $(1 - \ell_i(\Delta t)) \cdot \alpha_i$ represents the power of the node i available for the execution of grid tasks.

As regards the resources requested by the application task, we assume to know for each task k the number of instructions γ_k and the amount of communications ψ_{km} between the k -th and the m -th task $\forall m \neq k$ to be executed. All this information can be obtained either by a static program analysis or by using smart compilers or by other tools which automatically generate them. For instance the Globus Toolkit includes an XML standard format to define application requirements [12].

Encoding. Any mapping solution S is represented by a vector $\mu = (\mu_1, \dots, \mu_P)$ of P integers ranging in the interval $[0, N - 1]$, where the value $\mu_i = j$ contained in the i -th position means that the solution S under consideration maps the i -th task of the application onto node j of the grid.

Fitness. The fitness accounts for the time of use of resources and should be minimized. Denoting with $\theta_{ij}^{\text{comp}}$ and $\theta_{ij}^{\text{comm}}$ respectively the computation and the communication times requested to execute the task i on the node j it is assigned to, the total time needed to execute the task i on the node j is:

$$\theta_{ij} = \theta_{ij}^{\text{comp}} + \theta_{ij}^{\text{comm}}$$

It is computed on the basis of the computation power and of the bandwidth effectively available on that node. In fact, once deducted the resources necessary to execute the local workload, θ_{ij} is evaluated taking into account the concurrent execution of the task i and of all the other tasks assigned to the same node j . Obviously, this introduces a predetermined interaction among variables, i.e., the tasks mapped on the same node, while the communications introduce an interaction among tasks mapped on different nodes in the grid. In other words, a change in a variable influences the adaptation of the correlated ones.

It is to note that, given a mapping solution, the total time used by the node j to execute all the tasks assigned to it is represented by the maximum θ_{ij} .

Setting with $\phi_i \equiv \phi(i, \mu_i) = \theta_{ij}$ the fitness function assigned to the i -th task assigned to the node j , the fitness function of the mapping solution is:

$$\Phi(\boldsymbol{\mu}) = \max_{i \in [1, P]} \{\phi_i\} \quad (1)$$

This innovative definition of the above parameter-wise objective function aims to search for the smallest fitness value among these maxima, i.e. to find the mapping which uses at the minimum, in terms of time, the grid resource it has to exploit at the most.

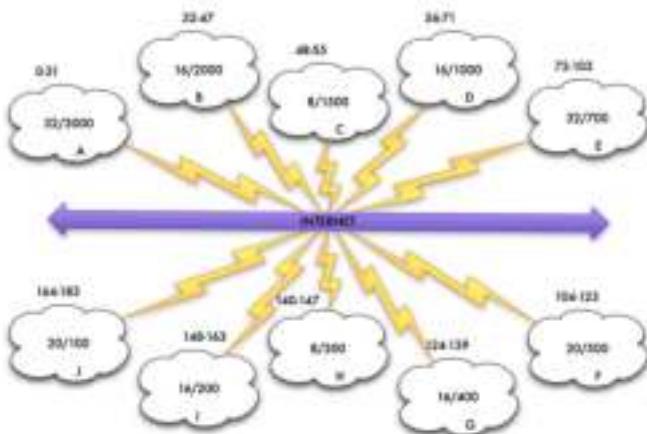
It is worth noting that, since all the tasks are co-scheduled (tasks contemporaneously loaded in the queues of runnable processes of the nodes which they are assigned to), if they run in perfect overlapping conditions, the time required to complete the application execution is given by the maximum value among all the θ_{ij} . In general this time ranges within the above smallest fitness value and $\sum_{i \in [1, P]} \phi_i$.

Furthermore, it is interesting to point out that, from the coevolutionary point of view, eq. (1) corresponds precisely to the *criticality threshold* delineated in the Bak-Sneppen model [10].

4 Simulation Results

To set up our simulational framework we assume to have a multisite grid architecture composed of ten sites containing a total of $N = 184$ nodes which are indicated by means of the external numbers from 0 to 183 in Fig. 1. For example 57 is the second of the 16 nodes in the site D . The values for the computing capabilities, the communication bandwidths and the load conditions of the grid nodes should be chosen conveniently because it is in this way easier, by arranging suitably simulations, to know which the optimal solutions should be and thus assess the goodness of the solutions achieved by τ -EO.

Without loss of generality we suppose that all the nodes belonging to the same site have the same power α expressed in terms of millions of instructions per second (MIPS). For example all the nodes belonging to E have $\alpha = 700$. As concerns the bandwidth we denote with β_{ii} the one for two communicating tasks being executed on the same node i , and with β_{ij} that for the two tasks being executed one on the node i and the other on the node j . The former

**Fig. 1.** The grid architecture**Table 1.** Intersite and intrasite bandwidths expressed in Mbit/s

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>	<i>J</i>
<i>A</i>	100									
<i>B</i>	2	200								
<i>C</i>	2	2	200							
<i>D</i>	4	4	4	400						
<i>E</i>	4	4	4	4	400					
<i>F</i>	8	8	8	8	8	800				
<i>G</i>	16	16	16	16	16	16	1000			
<i>H</i>	16	16	16	16	16	16	16	1000		
<i>I</i>	32	32	32	32	32	32	32	32	2000	
<i>J</i>	32	32	32	32	32	32	32	32	2000	

represents the *intranode* communication, whereas the latter denotes either the *introsite* communication (when the nodes i and j belong to the same site) or the *intersite* communication (when the nodes i and j belong to different sites). For the sake of simplicity we presume that $\beta_{ij} = \beta_{ji}$ and that all the β_{ij} s have the same very high value (100 Gbit/s) so that the related communication time is negligible with respect to intrasite and intersite communications (Table 1).

Since a generally accepted set of heterogenous computing benchmarks does not exist, to evaluate the effectiveness of our τ -EO approach we have conceived and explored several scenarios differing in number of tasks, amount of computations and communications and loads of the nodes. With the aim to discuss the behavior of the mapper, four cases have been reported among the numerous simulations effected: one for $P = 60$ and the other ones for $P = 30$.

After a preliminary tuning phase N_{iter} parameter has been set to 200,000 and τ to 3.0. For each problem 20 τ -EO runs have been performed, differing in the seeds for the random number generator. Henceforth we shall denote by μ_{Φ} the best solutions found in terms of lowest maximal resource utilization time.

The results and the solutions achieved for all the tests are summarized in Table 2. In it Φ^b denotes the best fitness value for any test. Also the average $\langle \Phi \rangle$ and the standard deviation σ_Φ computed over the 20 runs are shown. The best solution will be presented explicitly for the first simulation only, whereas, due to brevity's sake, for the other simulations they will be reported concisely as a list whose generic component has the format: $S(\mathcal{P} : \mathcal{T})$ where S is the site, \mathcal{P} the number of nodes used in that site and \mathcal{T} the number of there allocated tasks.

Simulation 1. It regards an application of $P = 60$ tasks divided into three groups \mathcal{G}_1 , \mathcal{G}_2 and \mathcal{G}_3 of 20 tasks each. Each task in \mathcal{G}_1 has $\gamma_k = 90,000$ MI, communicates only with all those in \mathcal{G}_3 with $\psi_{km} = 100$ Mbit $\forall m \in [41, \dots, 60]$. Each one in \mathcal{G}_2 has $\gamma_k = 900$ MI, communicates only with all those in \mathcal{G}_3 with $\psi_{km} = 1000$ Mbit $\forall m \in [41, \dots, 60]$. Each one in \mathcal{G}_3 has $\gamma_k = 90$ MI, communicates also with all the others in the same group with $\psi_{km} = 3,000$ Mbit $\forall m \in [41, \dots, 60]$. Moreover $\ell_i(\Delta t) = 0.0$ for all the nodes.

This problem is quite balanced between computation and communication and τ -EO should find a solution distributing all the tasks to nodes belonging to sites able to satisfy either computation or communication requirements, i.e. those of A and I respectively. The best mapping provided is the ideal one:

$$\boldsymbol{\mu}_\Phi = \{0, 2, 4, 6, 18, 7, 3, 23, 12, 22, 27, 19, 11, 24, 15, 29, 8, 30, 21, 9, 155, 148, 162, 160, 148, 154, 149, 152, 148, 162, 152, 160, 163, 151, 163, 161, 154, 155, 156, 153, 159, 152, 150, 156, 157, 151, 149, 153, 163, 161, 158, 162, 155, 154, 160, 161, 149, 153, 151, 156\} = \{A(20 : 20); I(16 : 40)\}$$

with $\Phi = 246.04$ s. The presence of sustained computation for some tasks and sustained communication for some others has a very strong influence on the best solution, which selects some nodes in A for the computation-bound tasks and all the nodes in the site I for the communication-bound ones.

Simulation 2. An application of $P = 30$ tasks is divided into two groups \mathcal{G}_1 and \mathcal{G}_2 of 15 tasks each. Each task has $\gamma_k = 90,000$ MI. Moreover, the generic task i in \mathcal{G}_1 communicates only with the task $j = (i + 15)$ in \mathcal{G}_2 with $\psi_{ij} = 100$ Mbit $\forall i \in [0, \dots, 14]$. As concerns the average load we have: $\ell_i(\Delta t) = 0.0 \ \forall i \in [0, \dots, 19]$, $\ell_i(\Delta t) = 0.5 \ \forall i \in [20, \dots, 31]$, $\ell_i(\Delta t) = 0.0 \ \forall i \in [32, \dots, 41]$, $\ell_i(\Delta t) = 0.5 \ \forall i \in [42, \dots, 47]$, $\ell_i(\Delta t) = 0.0 \ \forall i \in [48, \dots, 183]$.

This simulation requires that each task in the 15 pairs, in addition to communicating 100 Mbit with the related partner, performs 90,000 MI. In this hypothesis, the optimal allocation entails both the use of the most powerful nodes and the distribution of the communicating tasks in pairs on the same site. This is what happens in the solutions found: $\boldsymbol{\mu}_\Phi = \{A(22 : 22); B(8 : 8)\}$ with $\Phi = 45.50$ s. In fact given that, due to the loads, the most powerful nodes are the first 24 of A and the first 10 of B , the solution allocates 11 task pairs on 22 unloaded nodes in A and the remaining 4 pairs on 8 unloaded nodes in B . It is to note that such a solution requires $\Phi = 45.50$ s obtained by adding the computation time to execute 90,000 MI on an unloaded node of B ($90,000\text{MI}/2,000\text{MIPS} = 45$ s) and the communication time to send 100

Mbit between the tasks of the same pair arranged on unloaded nodes of B ($100\text{Mbit}/200\text{Mbit/s} = 0.5\text{s}$).

Simulation 3. In this simulation we have $P = 30$ tasks as above, but the tasks are divided into the following 7 groups: $\mathcal{G}_1 = \{0, \dots, 2\}$, $\mathcal{G}_2 = \{3, \dots, 11\}$, $\mathcal{G}_3 = \{12, \dots, 14\}$, $\mathcal{G}_4 = \{15, \dots, 17\}$, $\mathcal{G}_5 = \{18, 19\}$, $\mathcal{G}_6 = \{20, \dots, 22\}$ and $\mathcal{G}_7 = \{23, \dots, 29\}$. The tasks in \mathcal{G}_1 and \mathcal{G}_4 have $\gamma_k = 900$ MI, the tasks in \mathcal{G}_2 , \mathcal{G}_5 and \mathcal{G}_7 have $\gamma_k = 90,000$ MI, while those in \mathcal{G}_3 and \mathcal{G}_6 have $\gamma_k = 90$ MI. Moreover, only the pairs of tasks $(0, 15)$, $(1, 16)$ and $(2, 17)$ belonging to \mathcal{G}_1 and \mathcal{G}_4 communicate with $\psi = 1000$ Mbit, the tasks belonging to \mathcal{G}_2 , \mathcal{G}_5 and \mathcal{G}_7 have no mutual communications, and only the pairs of tasks $(12, 20)$, $(13, 21)$ and $(14, 22)$ belonging to \mathcal{G}_3 and \mathcal{G}_6 communicate with $\psi = 10,000$ Mbit. For the average load we have: $\ell_i(\Delta t) = 0.33 \forall i \in [0, \dots, 7]$, $\ell_i(\Delta t) = 0.9 \forall i \in [8, \dots, 31]$, $\ell_i(\Delta t) = 0.5 \forall i \in [32, 33]$, $\ell_i(\Delta t) = 0.9 \forall i \in [34, \dots, 47]$, $\ell_i(\Delta t) = 0.0 \forall i \in [48, 49]$, $\ell_i(\Delta t) = 0.9 \forall i \in [50, \dots, 55]$, $\ell_i(\Delta t) = 0.0 \forall i \in [56, \dots, 59]$, $\ell_i(\Delta t) = 0.5 \forall i \in [60, \dots, 103]$, and $\ell_i(\Delta t) = 0.0 \forall i \in [104, \dots, 183]$.

In the above load conditions, the most powerful nodes are 48 and 49 of C which, having a null load, are able to execute 1500 MIPS and the first 8 nodes of A which, being loaded at 33%, have a remaining power of 2010 MIPS. In addition, we have also some nodes in other sites with an available power of 1000 MIPS. Since the application has 18 non-communicating tasks which have to execute 90,000 MI, it is evident that, mapping 16 of the 18 tasks on the first 8 nodes in A and 2 on the 2 unloaded nodes in C , the computation time is about 90s. As it can be noted, the arrangement of the remaining 12 tasks so as to avoid both intersite and intrasite communications yields that their computation time is less than 90s. The best deployment we have obtained is $\mu_\Phi = \{A(12 : 20); C(2 : 2); D(1 : 1); E(5 : 5); J(2 : 2)\}$ with $\Phi = 89.55\text{s}$. According to such a solution, all the 18 most expensive tasks are allocated on the most powerful nodes (16 of A and 2 of C), while all the remaining ones are arranged in such a way to avoid both intersite and intrasite communications, thus achieving the optimum.

Simulation 4. It is like the former one except that the average node loads are randomly generated in $[0, 1]$. Such a simulation is the most complex in that, due to the random load of the grid nodes, we have no idea about the optimal allocation. As a consequence, we can only argue about some possibilities. Anyway, even if we had all the nodes in A totally unloaded, the best solution could not have Φ lower than $90,000\text{MI}/3,000\text{MIPS} = 30\text{s}$. On the other hand, given that we have generated the loads at random, we expect that on average 16 nodes in A will have a load greater than 0.5 and 16 a load lower than 0.5. In this state, the solution should have $\Phi = 90,000\text{MI}/1,500\text{MIPS} = 60\text{s}$. Anyway, the best solution found by τ -EO is very close to the latter situation:

$$\mu_\Phi = \{A(16 : 16); B(4 : 4); D(1 : 1); E(3 : 3); G(2 : 2); H(2 : 2); J(2 : 2)\}$$

with $\Phi = 60.71\text{s}$. It is interesting to note from the results in Table 2 that for all the experiments the average final values $\langle\Phi\rangle$ are very close to the best ones found Φ^b , and that the related standard deviation σ_Φ is quite low. This shows that the

Table 2. Simulation results

<i>Sim. No.</i>	1	2	3	4
	A(20:20);I(16:40)	A(22:22);B(8:8)	A(12:20);C(2:2); D(1:1);E(5:5); J(2:2)	A(16:16);B(4:4); D(1:1);E(3:3); G(2:2);H(2:2); J(2:2)
$\bar{\Phi}^b$	246.04	45.50	89.55	60.71
$\langle \Phi \rangle$	256.15	45.50	89.66	61.34
σ_Φ	4.83	0.00	0.20	1.29

algorithm does not depend too much on the randomly set initial solution, so we are confident that it is robust.

5 Conclusions

Extremal Optimization has been here proposed as a viable approach to carry out the mapping of the tasks making up a user application in grid environments. The idea behind such an approach is that the problem shows correlations among variables due to both communications and multitasking executions. So, the use of a coevolutionary approach that allows a parameter-wise evaluation of the objective function is possible.

The proposed method has been tested on a set of distinct simulations differing in number of application tasks to be mapped, quantity of computation, presence of communication and load of the grid nodes. These cases represent a suitable set of typical real-time situations, and for those in which the solution was *a priori* known, the tool has provided it. EO shows two very interesting features when compared to other optimization tools based on other Evolutionary Algorithms like Differential Evolution, also implemented in [16]. The first feature consists in a higher speed of about 3 times, while the second is its ability to provide stable solutions. However, due to the lack of systems which have same conditions as ours, a comparison to ascertain the effectiveness of our multisite mapping approach is difficult. In fact some of these algorithms, such as Minmin, Max min, XSuffrage, are related to independent subtasks and their performance are affected in heterogenous environments. In case of dependent tasks, the classical approaches use Direct Acyclic Graph differently from ours in which no assumptions are made about the communication timing among the processes.

References

1. Mateescu, G.: Quality of service on the grid via metascheduling with resource co-scheduling and co-reservation. International Journal of High Performance Computing Applications 17(3), 209–218 (2003)
2. Fernandez-Baca, D.: Allocating modules to processors in a distributed system. IEEE Transactions on Software Engineering 15(11), 1427–1436 (1989)

3. Wang, L., Siegel, J.S., Roychowdhury, V.P., Maciejewski, A.A.: Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *Journal of Parallel and Distributed Computing* 47, 8–22 (1997)
4. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing* 61, 810–837 (2001)
5. Kim, S., Weissman, J.B.: A genetic algorithm based approach for scheduling decomposable data grid applications. In: International Conference on Parallel Processing (ICPP 2004), Montreal, Quebec, Canada, pp. 406–413 (2004)
6. Song, S., Kwok, Y.K., Hwang, K.: Security-driven heuristics and a fast genetic algorithm for trusted grid job scheduling. In: IPDP 2005, Denver, Colorado (2005)
7. Boettcher, S., Percus, A.G.: Extremal optimization: an evolutionary local-search algorithm. In: Bhargava, H.M., Ye, N. (eds.) *Computational Modeling and Problem Solving in the Networked World*. Kluwer, Boston (2003)
8. Boettcher, S., Percus, A.G.: Extremal optimization: methods derived from co-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 825–832. Morgan Kaufmann, San Francisco (1999)
9. Dong, F., Akl, S.G.: Scheduling algorithms for grid computing: state of the art and open problems. Technical Report 2006–504, School of Computing, Queen’s University Kingston, Ontario, Canada (2006)
10. Sneppen, K., Bak, P., Flyvbjerg, H., Jensen, M.H.: Evolution as a self-organized critical phenomenon. *Proc. Natl. Acad. Sci.* 92, 5209–5213 (1995)
11. Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., Tuecke, S.: A directory service for configuring high-performance distributed computations. In: Sixth Symp. on High Performance Distributed Computing, Portland, OR, USA, pp. 365–375. IEEE Computer Society, Los Alamitos (1997)
12. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid information services for distributed resource sharing. In: Tenth Symp. on High Performance Distributed Computing, San Francisco, CA, USA, pp. 181–194. IEEE Computer Society, Los Alamitos (2001)
13. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
14. Wolski, R., Spring, N., Hayes, J.: The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems* 15(5–6), 757–768 (1999)
15. Gong, L., Sun, X.H., Waston, E.: Performance modeling and prediction of non-dedicated network computing. *IEEE Trans. on Computer* 51(9), 1041–1055 (2002)
16. De Falco, I., Della Cioppa, A., Maistro, D., Scafuri, U., Tarantino, E.: Multisite mapping onto grid environments using a multi-objective differential evolution. In: Differential Evolution: Fundamentals and Applications in Engineering, ch. 11. John Wiley, Chichester (to appear)

Swarm Intelligence Inspired Multicast Routing: An Ant Colony Optimization Approach*

Xiao-Min Hu, Jun Zhang**, and Li-Ming Zhang

Department of Computer Science, Sun Yat-Sen University, Guangzhou, P.R. China
junzhang@ieee.org

Abstract. The advancement of network induces great demands on a series of applications such as the multicast routing. This paper firstly makes a brief review on the algorithms in solving routing problems. Then it proposes a novel algorithm called the distance complete ant colony system (DCACS), which is aimed at solving the multicast routing problem by utilizing the ants to search for the best routes to send data packets from a source node to a group of destinations. The algorithm bases on the framework of the ant colony system (ACS) and adopts the Prim's algorithm to probabilistically construct a tree. Both the pheromone and heuristics influence the selection of the nodes. The destination nodes in the multicast network are given priority in the selection by the heuristics and a proper reinforcement proportion to the destination nodes is studied in the case experiments. Three types of heuristics are tested, and the results show that a modest heuristic reinforcement to the destination nodes can accelerate the convergence of the algorithm and achieve better results.

1 Introduction

With the development of network technologies, application demands and quality requests are getting higher and higher. As the topologies of networks are complex, efficient packet routing methods are quite meaningful to the real world [1].

The complexity of the network makes deterministic algorithms incapable to react in time. So a variety of heuristic algorithms for network routing emerged in recent years. Di Caro and Dorigo [2] introduced an approach called AntNet to the adaptive learning of routing tables in a communication network. Gelenbe et al. [3] recently proposed an experimental investigation of path discovery using genetic algorithms (GAs). There are also some algorithms for network routing, such as some agent-based algorithms (eg., the ant-based control (ABC) system [4]), neural heuristics [5]. These algorithms are promising but still need enhancements.

Multicast routing [6]-[11] first appeared in ARPANET in packet-switched store-and-forward communication networks. The packets in multicast routing

* This work was supported by NSFC Joint Fund with Guangdong, Key Project No. U0835002, NSF of China Project No.60573066 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, P.R. China.

** Corresponding author.

are sent from a source node to several destinations without or with certain constraints such as the requirements of the quality-of-service (QoS) and real-time communications. The unconstrained multicast routing problem, which is known to be NP-complete [10], is equivalent to the Steiner tree problem in graphs.

This paper aims at solving the unconstrained multicast routing problem using an ant algorithm. The proposed algorithm, which is called the distance complete ant colony system (DCACS), bases on an ant colony system framework [12] and takes advantage of the distance complete graph (DCG) to find the multicast tree with the lowest cost in the network. Each ant starts from a randomly chosen destination nodes (including the source node) and follows the structure of the Prim's minimum spanning tree (MST) [13] algorithm to construct the multicast tree. The ants select a next node in a deterministic way, or explore one by a probabilistic selection. After an ant has finished its tour, the solution is checked further for a potentially better tree by the classical Prim's MST algorithm and the redundancy trimming strategy. The pheromone in the network is updated by the local and global pheromone update mechanism.

The genetic algorithm presented in [10] and the ant algorithm described in [15] are used to compare the performance of the proposed DCACS algorithm. The ant algorithm in [15] has been applied to solve Steiner tree problems. However, their algorithm differs from ours in which: 1) Their algorithm bases on the ant system (AS), but ours on the ant colony system (ACS). 2) The ants in their algorithm start from all the destination nodes and merge together when an ant steps on the route that another ant has passed. Our ants start from a randomly chosen destination node, and each ant builds their own tree. 3) Their algorithm has conflict detection and avoidance mechanism as their ants may conflict with each other resulting in erroneous results. But ours do not need to do so. 4) The pheromone update methods are different between the two algorithms.

The other contribution of this paper is that it analyzes the results of giving priority in the selection of destination nodes by the heuristics. Three types of heuristic methods are tested. Proper heuristic methods and parameters settings to reinforce pheromone values to the destination nodes and the other nodes are concluded by experiments on the Steiner cases from the OR-Library [14].

The paper is organized as follows. Section 2 introduces the description of multicast routing problems and their mathematical definition. The construction method of the distance complete graph (DCG) is also introduced. Section 3 describes the implementation and main techniques of the proposed DCACS algorithm. In Section 4, the proposed algorithm is tested by some cases to analyze its performance. The conclusion of the paper is made in Section 5.

2 Multicast Routing Problem and Distance Complete Graph

In this section, the formal description and definition of the multicast routing problem will be made and the distance complete graph topology on which the proposed algorithm based will also be described.

2.1 Definition of the Multicast Routing Problem

A network graph is denoted as $G = (V_G, E_G, \Omega)$, where V_G is the set of nodes in the network, E_G is the set of edges that connect two nodes in V_G , and Ω is the set of constraints. If the problem is constraint-free, then $\Omega = \emptyset$. In a multicast routing problem, the V_G is divided into three subsets as V_S , V_T , V_I , where V_S is the set of source nodes, V_T is the set of destinations, and V_I is the set of intermediate nodes. Each edge e which belongs to V_G has a positive cost $c(e)$ for passing the edge. If there is no edge connecting two nodes in V_G , the corresponding cost will be set as ∞ , which indicates that the edge does not exist. The multicast routing problem is to find a tree with the minimum cost connecting the nodes in V_S and V_T , and the resulting tree must comply with the constraints in Ω . The formal definition of a multicast routing problem is to find a tree $T = (V_S + V_T + V_\theta, E_\theta)$, where $V_\theta \subseteq V_I$ and $E_\theta \subseteq E_G$, with

$$\min \sum_{e \in T} c(e), T \text{ satisfies constraints in } \Omega.$$

Fig. 1 shows an example of the multicast routing. There are twelve nodes and some edges. The gray solid node 1 is the source node, and the black solid nodes 2, 6, and 11 are the destination nodes. Now the problem is to find a tree which has all the destination nodes connected to the source node with the minimum total cost within the constraints. The black edges between nodes 2, 1, 5, 6, 7, 11 construct the result tree in the example via the intermediate nodes 5 and 7. As the problem considered in this paper is the one without constraints, which is an equivalence of the Steiner tree in graphs, we do not differentiate the source nodes and the destination nodes. We call both of the nodes as destination nodes. The sets V_S and V_T are combined to be denoted as V_D .

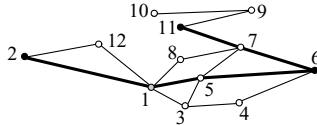


Fig. 1. An example of a multicast routing network

2.2 Distance Complete Graph

The network in multicast routing is not a fully-connected graph, which means that some nodes are not directly connected by a single edge, though they are attainable via other nodes and the corresponding edges. A distance complete graph (DCG) of a network is the one that each pair of nodes is logically connected and the cost (or distance) on that edge is the cost of the shortest path between the nodes in the actual network.

One of the classical algorithms for generating an undirected DCG is the Floyd algorithm [16]. Fig. 2 shows an example of a topology with six nodes. The solid edges in Fig. 2 are the actual edges existing between the two nodes, and the numbers beside the edges are the corresponding cost values. We can see that

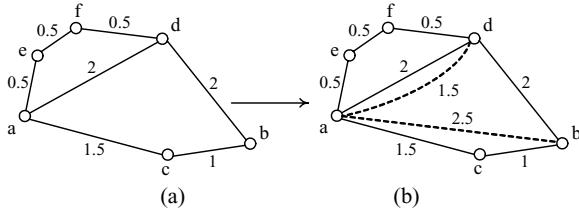


Fig. 2. An example of transforming edges (a,b) and (a,d) in the DCG

nodes a and b are not adjacent, but they can reach each other via node c or d. The algorithm is to change the original graph in Fig. 2(a) to the DCG in Fig. 2(b), where a logical edge (shown as a dashed line) is added with the minimum cost from a to b. The shortest route from a to b is via node c, so the cost d_{ab} of the logical edge in the DCG is 2.5. Because the graph is undirected, we have $d_{ba} = d_{ab}$. To nodes a and d, although they are connected directly, there is a shorter route via nodes e and f. So $d_{ad} = d_{da} = 1.5$.

The time complexity of the Floyd algorithm is $O(|V_G|^3)$. The ant algorithm proposed in this paper is based on the DCG, so the construction of the DCG must be preprocessed. After constructing the DCG, the ants can be dispatched to construct multicast trees.

3 Distance Complete ACS for Multicast Routing Problem

The structure of the proposed distance complete ACS (DCACS) is based on the procedure of constructing a minimum spanning tree (MST) by the Prim's algorithm [13]. The following section will present how to solve a multicast routing problem.

3.1 The Prim's Algorithm and the Pheromone and Heuristics Mechanism

Suppose S is a set with one node and V is the set of nodes in an undirected connected graph excluding the node in S . The classical Prim's algorithm [13] is to add the node j to S , provided that the edge (i, j) has the minimum cost, $i \in S$, and $j \in V - S$. The algorithm terminates when $S = V$. In the proposed algorithm, the selection criteria for the next node are not simply based on the cost of the edges, but the product of the pheromone value and the heuristic value.

The product of pheromone and heuristics for an edge (i, j) is denoted as $\tau(i, j)\eta(i, j)$, where $\tau(i, j)$ is the pheromone value and $\eta(i, j)$ is the heuristic value. The initial pheromone value τ_0 for every edge is discussed in 3.2. Three types of heuristic values are tested in this paper.

$$H_a : \eta(i, j) = \begin{cases} [\mu_D / (1/d_{ij})]^\beta, & \text{if } j \in V_D \\ [\mu_I / (1/d_{ij})]^\beta, & \text{otherwise} \end{cases} \quad (1)$$

$$H_b : \eta(i, j) = \begin{cases} \mu_D / (1/d_{ij})^\beta, & \text{if } j \in V_D \\ \mu_I / (1/d_{ij})^\beta, & \text{otherwise} \end{cases} \quad (2)$$

$$H_c : \eta(i, j) = \begin{cases} (\mu_D)^\beta / (1/d_{ij}), & \text{if } j \in V_D \\ (\mu_I)^\beta / (1/d_{ij}), & \text{otherwise} \end{cases} \quad (3)$$

where μ_D and μ_I are the reinforcement proportion to the destinations and intermediate nodes respectively. $\mu_D = \max(|V_I|/|V_G|, |V_D|/|V_G|)$. $\mu_I = \min(|V_I|/|V_G|, |V_D|/|V_G|)$. d_{ij} is the cost of the logical edge in the DCG between nodes i and j . The parameter β is a positive constant.

The destination nodes are more likely to be selected by using the above heuristic methods. With the same cost by the edges, the greedy operation prefers the nodes in V_D . The performance of the algorithm with different heuristic methods will be tested in Section 4 with a discussion on the design of the heuristics.

3.2 The Ant's Search Behavior

In this part, the process on how an ant searches for routes to build a tree is described step by step. It is a probabilistic Prim's algorithm in the framework of ACS [12] and the algorithm operates in the DCG.

Step 1: Initialization

At first, an ant k is placed on a randomly chosen destination node i . Then it begins its search.

Step 2: Exploration or Exploitation

The ant probabilistically chooses to do exploration or exploitation to select the next node j based on the state transition rule [12], which is controlled by (4)

$$j = \begin{cases} \arg \max_{r \in \Theta_k} \{\tau(i, r)\eta(i, r)\}, & \text{if } q \leq q_0 \text{ (exploitation)} \\ J, & \text{otherwise (exploration)} \end{cases} \quad (4)$$

where i is the node that the ant has visited, Θ_k is the set of nodes that haven't been visited by the current ant k , q_0 is a predefined parameter in $[0, 1]$ controlling the proportion of doing exploitation and exploration, J is a random node chosen by (5). $\tau(i, r)$ stands for the pheromone value on the edge connecting nodes i and r . $\eta(i, r)$ is the heuristic value for choosing node r from node i . Note that the DCG is constructed, so any pairs of nodes are logically connected. If a random number q is smaller than or equal to q_0 , the ant will choose the next unvisited node with the maximum product of pheromone and heuristic value. This is called the exploitation step. Otherwise, the next node j is chosen by (5) with a probability distribution, which is called the exploration step.

$$p_k(i, j) = \begin{cases} \frac{\tau(i, j)\eta(i, j)}{\sum_{r \in \Theta_k} \tau(i, r)\eta(i, r)}, & \text{if } j \in \Theta_k \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Step 3: Edge Extension and Local Pheromone Update

When an ant is building a solution, the pheromone value on the visited edges is decreased. It is done for the avoiding too many ants that tail after the same route

and letting the other ants search the network diversely. The local pheromone update is done in two sub-steps, one is the update of pheromone on logical edges, and the other is on actual edges. The pheromone value on an edge (i, j) is updated as (6).

$$\tau(i, j) = (1 - \rho)\tau(i, j) + \rho\tau_{\min} \quad (6)$$

where $\rho \in (0, 1]$ is the pheromone evaporation rate. τ_{\min} is the lower boundary value of the pheromone on every edge. Since $\tau(i, j) \geq \tau_{\min}$, the updated pheromone value is smaller than or equal to the original one.

After choosing the next node j , an ant k moves from node i to node j . Since the ant moves on the DCG, the logical edge (i, j) can be extended to the actual route. The logical and actual edges in the route have their pheromone updated by (6). Fig. 3 shows an example of updating an edge $(1, 6)$, which means that the ant just moved from node 1 to node 6. The logical edge $(1, 6)$ can be extended to be a route $1, 2, 3, 4, 5, 6$. Then the pheromone values on the corresponding edges $(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (5, 6)$ and their symmetric edges (e.g. $(2, 1)$) are also updated. After the local pheromone update, the pheromone density on the edges corresponding to the logical edges decreases.

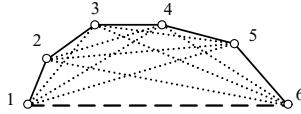


Fig. 3. Example of extending a logical edge $(1, 6)$

Step 4: Has the ant finished the job?

The nodes walked by the ant and the ones that are extended from the logical edges are regarded as having been visited. The termination condition for an ant is whether it has visited all the destination nodes. If the termination condition is unsatisfied, go to Step 2 for a further search. Otherwise, the ant has finished searching for a multicast tree.

3.3 Redundancy Trimming

After an ant has finished building a multicast tree connecting the source node and all the destination nodes, the tree must be checked for redundancy.

Firstly, we apply the classical Prim's algorithm to the visited nodes. If the cost of the generated MST is smaller than that of the tree built by the ant, the ant's solution is replaced by the MST.

Secondly, we check for useless intermediate nodes. The intermediate nodes that have only one degree of connectivity are deleted (e.g. node 10 will be deleted in Fig. 1).

The above two processes are repeated until the tree cannot be optimized any more. The redundancy trimming can also be used as a deterministic algorithm

for multicast routing problems. Take the whole set of nodes in V_G as the input, the tree can be gradually trimmed and reduced. The resulting tree is a MST, and it is a feasible solution for the problem. However, the experiments in Section 4 show that this method can not generate the best tree. It is a sub-optimal result, which can be used to initialize the pheromone value. Hence, the initial pheromone value on every edge is set as

$$\tau_0 = 1/(|V_G|T_s) \quad (7)$$

where T_s is the cost of the tree generated by the deterministic redundancy trimming method.

3.4 Global Pheromone Update

After the m ants have performed the tree construction, the global pheromone update is applied to the best tree ever been found in order to reinforce the pheromone values. The pheromone values on the actual edges of the best-so-far tree are updated as (8).

$$\tau(i, j) = (1 - \rho)\tau(i, j) + \rho\Delta\tau \quad (8)$$

where $\Delta\tau = 1/T_{\text{best}}$, T_{best} is the total cost of the best-so-far tree.

The pheromone values on the logical edges in the best-so-far tree are also updated. Suppose an actual route between a pair of nodes i and j in the best tree is $(a_0, a_1, a_2, \dots, a_\psi)$, where $a_0 = i$ and $a_\psi = j$. ψ is the number of edges in the route, then the new pheromone value on the edge (i, j) equals to

$$\tau(i, j) = \sum_{l=0}^{\psi-1} \tau(a_l, a_{l+1})/\psi \quad (9)$$

4 Experiment and Discussions

The test cases used in this paper are the Steiner problems in group b from the OR-Library [14]. The eighteen problems are tabulated in Table 1, with the graph size from 50 to 100. In the table, $|V_G|$ stands for the number of nodes, and $|V_D|$ is the number of destination nodes. $|E_G|$ is the number of actual edges in the graph. OPT is the optimum of the problem. T_s is the cost of the tree generated by using only the redundancy trimming method. It can be seen that the value of T_s is not good compared to the best value.

The values of the parameters are set empirically as $q_0 = 0.9$, $\rho = 0.1$, $\tau_{\min} = \tau_0$. The maximum iteration number is set as 500, but once the ants have found the optimum, the algorithm terminates. The same combination of parameters values are tested for ten times independently.

Table 1. Eighteen Steiner b test cases

No.	$ V_G $	$ E_G $	$ V_D $	OPT	T_s	No.	$ V_G $	$ E_G $	$ V_D $	OPT	T_s	No.	$ V_G $	$ E_G $	$ V_D $	OPT	T_s
1	50	63	9	82	89	7	75	94	13	111	112	13	100	125	17	165	189
2	50	63	13	83	96	8	75	94	19	104	111	14	100	125	25	235	243
3	50	63	25	138	144	9	75	94	38	220	227	15	100	125	50	318	333
4	50	100	9	59	63	10	75	150	13	86	95	16	100	200	17	127	171
5	50	100	13	61	68	11	75	150	19	88	103	17	100	200	25	131	144
6	50	100	25	122	130	12	75	150	38	174	186	18	100	200	50	218	227

Table 2. Comparisons of DCACS with the GA in [10] and the ant algorithm in [15]

No.	GA[10]	ant[15]	DCACS	No.	GA[10]	ant[15]	DCACS	No.	GA[10]	ant[15]	DCACS
1	82	82	82	7	111	—	111	13	165	167.3	168
2	83	83	83	8	104	104	104	14	235	235.3	235
3	138	138	138	9	220	220	220	15	318	318	318
4	59	59	59	10	86	87.4	86	16	127	133	127
5	61	61	61	11	88	89	88	17	131	—	131
6	122	—	122	12	174	—	174	18	218	225.5	218

— The values are not available in [15].

4.1 Analysis on the Heuristic Method

The basic structures of H_a , H_b , and H_c are the same, except for the role of the parameter β . The reinforcement proportion to the destination nodes and the intermediate nodes is normalized to be in $[0, 1]$ and $\mu_D + \mu_I = 1$. In the traditional ACS for the traveling salesman problem by Dorigo and Gambardella [12], the heuristics is defined as $(1/c_{ij})^\beta$, and the β ($\beta > 0$) is used to determine the relative importance of pheromone versus cost. In our algorithms in solving multicast routing problems, a series of tests is made to judge in which way the parameter β should be used and what its value should be.

The results by the three H_a , H_b , and H_c are compared in Fig. 4. Each sub-figure is composed of three parts with the corresponding heuristic methods, and each method has been tested using 10, 50 and 100 ants. The figures illustrate the success rates of the algorithm to the corresponding Steiner tree problems. It can be seen that H_a and H_c are more robust than H_b , which means that the emphasis by β to μ_D and μ_I is more significant than to the cost of the edges. The best β for $H2_a$ is 1, 2, 3 and for $H2_c$ is 1, 2, 3, 4 in most cases. The best value of μ^β is shown to be from 8 to 16 in the test cases.

4.2 Comparison with Other Algorithms

We use the genetic algorithm in [10] and the ant algorithm in [15] to compare the performance of the proposed algorithm. The parameters used in the proposed DCACS with $H2_c$ are set as $m = 10 \sim 100$, $\beta = 3$, $\rho = 0.1$, and $q_0 = 0.9$.

The obtained results are tabulated and compared in Table 2. The genetic algorithm presented in [10] can find the optimal tree with 100% success to all

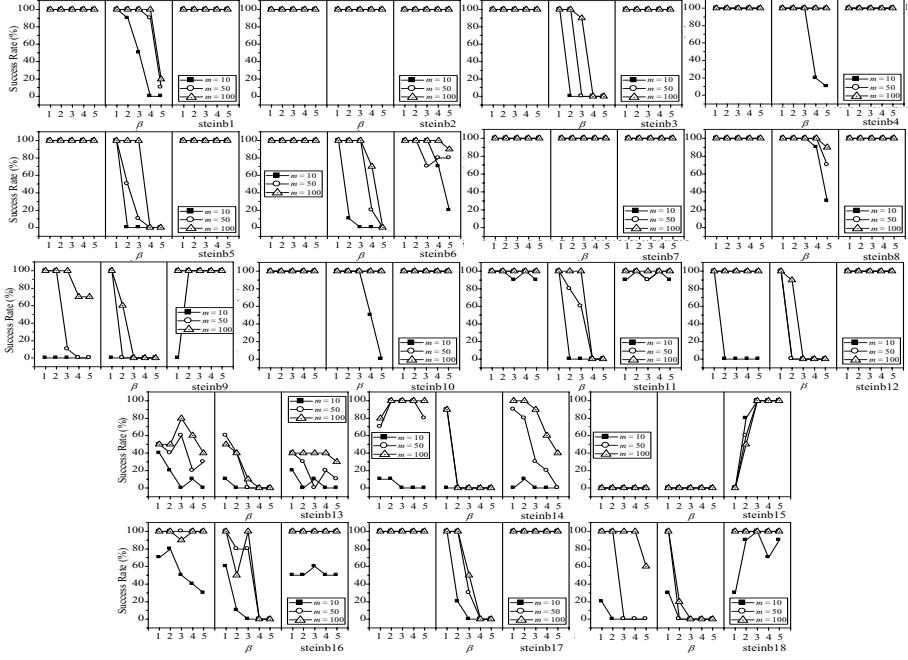


Fig. 4. Comparison between $H2_a$, $H2_b$ and $H2_c$

the eighteen test problems. Our proposed DCACS also can achieve 100% success to the test cases except for Stein b13. In [15], only the Stein b1~b5, b8~b11, b13~b16, and b18 are tested. However, the results obtained by our algorithm are much better and faster according to the descriptions in [15].

5 Conclusion

This paper proposed an algorithm DCACS according to the ant colony system on the distance complete graph to solve the multicast routing problems without constraints. The ants base on the structure of Prim's algorithm and probabilistically select the nodes to construct a multicast tree. After the tree has been generated, the redundant nodes are deleted. Pheromone update and heuristic reinforcement to the destination nodes efficiently guide the algorithm to the optimum. The proposed algorithm is tested by eighteen Steiner cases and three kinds of heuristic methods are evaluated. By comparing the results with other algorithms, the proposed algorithm is very promising in solving multicast routing problems.

References

1. Tanenbaum, A.: Computer Networks. Prentice-Hall, Englewood Cliffs (1996)
2. Di Caro, G., Dorigo, M.: Antnet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)

3. Gelenbe, E., Liu, P.X., Lain, J.: Genetic Algorithms for Autonomic Route Discovery. In: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and its applications, DIS 2006, pp. 371–376 (2006)
4. Schoonderwoerd, R., Holland, O., Bruton, J., Rothkrantz, L.: Ant-like Agents for Load Balancing in Telecommunications Networks. In: Proceedings of the First International Conference on Autonomous Agents (Agents 1997), Marina del Rey, CA, pp. 209–216 (1997)
5. Gelenbe, E., Ghanwani, A., Srinivasan, V.: Improved Neural Heuristics for Multicast Routing. IEEE Journal on Selected Areas in Communications 15(2), 147–155 (1997)
6. Salama, H.F., Reeves, D.S., Viniotis, Y.: Evaluation of Multicast Routing Algorithms for Real-time Communication on High-speed Networks. IEEE Journal on Selected Areas in Communications 15(3), 332–345 (1997)
7. Wang, B., Hou, J.C.: Multicast Routing and Its Qos Extension: Problems, Algorithms, and Protocols. IEEE Network 14(1), 22–36 (2000)
8. Low, C.P., Song, X.-Y.: On Finding Feasible Solutions for the Delay Constrained Group Multicast Routing Problem. IEEE Trans. on Computers 51(5), 581–588 (2002)
9. Charikar, M., Naor, J., Schieber, B.: Resource Optimization in QoS Multicast Routing of Real-time Multimedia. IEEE/ACM Trans. on Networking 12(2), 340–348 (2004)
10. Leung, Y., Li, G., Xu, Z.B.: A Genetic Algorithm for the Multiple Destination Routing Problems. IEEE Trans. on Evolutionary Computation 2(4), 150–161 (1998)
11. Bharath-Kumar, K., Jeffe, J.M.: Routing to Multiple Destination in Computer Networks. IEEE Trans. on Communication 31(3), 343–351 (1983)
12. Dorigo, M., Gambardella, L.M.: Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans. on Evolutionary Computation 1(1), 53–66 (1997)
13. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
14. Beasley, J.E.: OR-Library: Distributing Test Problems by Electronic Mail. J. Opl. Res. Soc. 41(11), 1069–1072 (1990)
15. Singh, G., Das, S., Gosavi, S., Pujar, S.: Ant Colony Algorithms for Steiner Trees: an Application to Routing in Sensor Networks. In: Recent Developments in Biologically Inspired Computing, pp. 181–206. Idea Group Publishing (2005)
16. Floyd, R.W.: Shortest Path. Communications of the ACM, 345 (1962)

A Framework for Evolutionary Peer-to-Peer Overlay Schemes

Michele Amoretti

Dipartimento di Ingegneria dell'Informazione, University of Parma,
Via Usberti 181/a, 43039 Parma, Italy
`michele.amoretti@unipr.it`

Abstract. In the last half-decade, many considerable peer-to-peer protocols have been proposed. They can be grouped in few architectural models, taking into account basically two dimensions: the dispersion degree of information about shared resources (centralized, decentralized, hybrid), and the logical organization (unstructured, structured). On the other side, there is a lack of common understanding about adaptive peer-to-peer systems. In our view, peers' internal structure may change in order to adapt to the environment, according to an adaptation plan. To formalize this approach, we propose the Adaptive Evolutionary Framework (AEF). Moreover, we apply it to the problem of sharing consumable resources, such as CPU, RAM, and disk space.

1 Introduction

Recently, the peer-to-peer (P2P) paradigm has emerged as a highly appealing solution for scalable and high-throughput resource sharing among decentralized computational entities, by using appropriate information and communication systems without the necessity for central coordination [16]. In distributed systems based on the peer-to-peer paradigm all participating processes have the same importance. In contrast with the client/server approach, in which resource providers and resource consumers are clearly distinct, on the contrary peers usually play both roles. Furthermore, a peer-to-peer network is a complex system, because it is composed of several interconnected parts (the peers) that as a whole exhibit one or more properties (*i.e.* behavior) which are not easily inferred from the properties of the individual parts [14]. The reaction of a peer to direct or indirect inputs from the environment is defined by its internal structure, which can be based either on static rules shared by every peer (protocols), or based on an adaptive plan τ which determines successive structural modifications in response to the environment, and turns the P2P network in a complex adaptive system (CAS) [9].

In the last half-decade, many considerable peer-to-peer protocols have been proposed. They can be grouped in few architectural models, taking into account basically two dimensions: the dispersion degree of information about shared resources (centralized, decentralized, hybrid), and the logical organization (unstructured, structured). The behavior of a peer-to-peer system based on protocols follows a pre-established pattern.

On the other side, there is a lack of common understanding about adaptiveness. In our view, illustrated by fig. 1, peers' internal structure may change in order to adapt to the environment. For example, consider a search algorithm whose parameters' values change over time in a different way for each peer depending on local performance indicators. The evolution of a structure can be *phylogenetic*, for which memoryless transformations are applied to the structure to modify it, or *cultural or epigenetic*, which assumes learning and knowledge transmission. In general, adaptive peer-to-peer networks emulate the ability of biological systems to cope with unforeseen scenarios, variations in the environment or presence of deviant peers.

Adaptation mechanisms running in background introduce additional costs in terms of local resources consumed and message traffic in the network. However, also the most advanced protocols of "traditional" peer-to-peer systems, *e.g.* Chord and Kademlia, require costly periodic self-maintenance, such as stabilization of routing tables, etc.

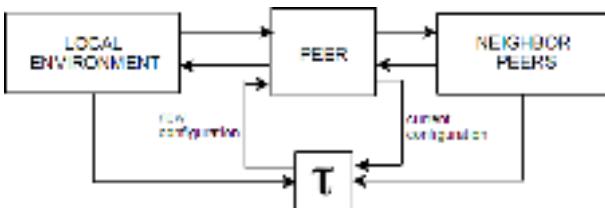


Fig. 1. Interactions between a peer, local environment and neighbors in an adaptive P2P architecture

In fig. 1 we make a distinction between local environment and neighbor peers. The local environment is what the peer perceives with its sensors: direct inputs from users, but also contextual information (this is the typical case of ambient intelligence scenarios). The peer and its neighbors are part of the P2P system which is immersed in the environment. The response of the peer to the inputs that come from the local environment is usually contingent on interactions with neighbors (which in turn may involve their neighbors, etc.). The other way round, a peer can receive requests from its neighbors, and its response in general may depend and affect its local environment.

In this paper we formalize our view by proposing the **Adaptive Evolutionary Framework (AEF)** for peer-to-peer architectures. In section 2 we define the theoretical foundation of AEF, with particular emphasis on adaptation based on genetic algorithms. To provide a concrete example, in section 3 we introduce the AEF strategy in a Gnutella-like overlay scheme for sharing consumable resources such as disk space, CPU cycles, etc. Related work on large-scale resource sharing architectures (*e.g.* Grids) is provided in section 4. Finally, section 5 concludes the paper summarizing its main results and proposing future work.

2 Adaptive Evolutionary Framework

The environment in which P2P networks operate is the set of users, robots, agents, etc. that directly or indirectly act on peers and on resources that each node can use and share with other nodes. Early P2P systems, both unstructured and structured, do not account any of the dynamics and heterogeneity of their environment. Systems like KaZaA and JXTA divide peers in supernodes and regular nodes, depending on the characteristics of their hosts, and provide strategies for dynamic promotions and demotions. Such kind of adaptation strategy may be affected by too much control overhead if the processing power available to peers, which depends not only on their hardware resources but also on usage load, quickly changes over time. Other systems incorporate IP network topology awareness into their design. An example is Pastry's proximity metric, for which each node key match in the routing table is tagged with the distance, in terms of IP hops, of the current node from the given IP address [15]. The drawback of this approach is its limited scope. A more comprehensive adaptation strategy has been proposed by Kalyvianki *et al.* [11], where each node analyzes log traces in order to find node capacity as perceived by the P2P application, by processing raw events such as frequency of schedule changes, number of running processes and duration of file system operations. Resulting information is periodically published in the overlay network, in order to be used by P2P applications to select nodes with best overall performance. The limit of this approach is the high bandwidth consumption due to control messaging, which could be excessive in case of high dynamism in the availability of each node's resources.

We propose the Adaptive Evolutionary Framework (AEF) as a response to these issues. According to the AEF, the internal structure of the peer is based on an adaptive plan τ which determines successive structural modifications in response to the environment. The adaptive plan, in the AEF framework, is based on an evolutionary algorithm, which utilizes a *population* of individuals (structures), where each individual represents a candidate solution to the considered problem.

Each structure C consists of p specified elements. The i th element ($i = 1, \dots, p$) of the structure has l_i possible values; the set of possible values for the i th element is defined as $A_i = \{a_{i1}, \dots, a_{il_i}\}$. Each structure represents a point in the search space \mathcal{A} , which is the domain of action of the adaptive plan. In other words, \mathcal{A} is the set of all combinations of values for the elements of the structure:

$$\mathcal{A} = A_1 \times A_2 \times \dots \times A_p \quad (1)$$

The adaptive plan τ produces a sequence of structures, *i.e.* a trajectory through \mathcal{A} , by emulating the process of natural evolution. Before the evolutionary process can start, an initial population (being it a subset of \mathcal{A}) must be generated. The size of the initial population has consequences for performance in terms of accuracy and the time to converge.

If all elements of a structure were binary, there would be 2^p possible structures. Considering the whole complex system, a network of N peers, we could state

that the number of possible configurations of the system is 2^{pN} . Fortunately, the environment usually affects a limited set of peers at a time. For example, a resource request is usually triggered by a user interacting with a peer, or by an application running on the same host (or in the same process space) of a peer which receives the request. We say that the peer is directly affected by the environment's input. The peer may own the requested resource, or may propagate the request to other peers. We say that these peers are indirectly affected by the environment's input. When the system receives an input, not all its nodes must be considered for evolution, but only those which are directly or indirectly affected by the environment's input. Thus, a reasonable tradeoff is to design τ plans at the level of single peer. Each node can evolve either autonomously or merging its structure with those of other nodes, *e.g.* of its k neighbors.

The total information received by τ up to generation g is given by the sequence $< I(0), I(1), \dots, I(g-1) >$, with $I \in \mathcal{I}$. The part of this information which is retained is the *memory*, and its domain is \mathcal{M} . Then the adaptive plan can be represented by

$$\tau : \mathcal{I} \times \mathcal{A}^+ \rightarrow \mathcal{A}^+ \quad (2)$$

where $\mathcal{A}^+ = \mathcal{A} \times \mathcal{M}$.

One case of particular importance is that in which the adaptive plan receives direct indication of the performance of each structures it tries. That is, a part of the input I is provided by a *fitness function*, which maps a structure representation into a scalar value:

$$F : \mathcal{A}^+ \rightarrow \mathbb{R} \quad (3)$$

The fitness function quantifies the quality of a structure, *i.e.* how close it is to optimality, with respect to the environment. It is therefore extremely important that the fitness function accurately models the problem, including all criteria to be optimized. Frequently the fitness function does not directly evaluate the elements of a structure, but their expression as physical features of behaviors. If the fitness function represents a cost (as usual), it should return lower values for better structures.

In addition to optimization criteria, the fitness function can also reflect the constraints of the problem through penalization of those structures that violate constraints. Constraints can be encapsulated within the fitness function, but also incorporated in the adaptation process.

Each run of an evolutionary algorithm produces a new generation of structures, representing a set of new potential configurations for the peer. The best structure of the new generation C_{g+1} is then selected from the offspring. Each new generation is formed through the application of a set of operators Ω to selected individuals of the current population. Selections are influenced by information obtained from the environment, so that the plan τ typically generates different trajectories in different environments.

To implement the AEF, different evolutionary strategies may be applied. The most obvious is using genetic algorithms (GAs) [10], for their relative simplicity,

but other solutions may be considered (for example, the *SLAC* algorithm proposed by Hales [8] can be placed under the AEF umbrella).

Until here we have used the term "structure" to indicate what the adactive plan modifies in an evolutionary peer. If the adative plan is a genetic algorithm, we can say that the structure is the code that maps the way the peer looks and/or acts (*phenotype*). In other words, the structure is the *genotype*, which is also called *chromosome*, in evolutionary computing (in genetics this simplification is not correct) [10], [5].

Each structure \mathbf{C} consists of p specified *genes* (in the general framework we used the generic term "elements"). The i th gene ($i = 1, \dots, p$) of the structure has l_i possible *alleles* (previously called "values"); the set of possible alleles for the i th element is $A_i = \{a_{i1}, \dots, a_{il_i}\}$, assuming finite values in limited specific ranges. Each structure represents a point in the search space \mathcal{A} , which is the set of all combinations of alleles (*i.e.* the *genome*). The pseudocode in Algorithm 1 describes the general GA scheme, assuming a set of W initial search points.

In GAs, most frequently used operators are reproduction (cross-over, mutation) and elitism, which act on genotypes. Throughout the adaptation process, the selection operator is also used, in order to emphasize best configurations (chromosomes) in a population. The interplay of these operators leads to system optimization. Whenever the entities reproduce they create a surplus, variation amounts to innovation of novel entities (*i.e.* reproduction is not simply cloning), and finally selection takes care of promoting the right variants by discarding the poor ones [4], [5].

Algorithm 1. General GA scheme

- 1: Let $g = 0$
 - 2: Define initial population $C_g = \{\mathbf{C}_{g,w} | w = 1, \dots, W\}$
 - 3: **while** termination criteria not fulfilled **do**
 - 4: Evaluate the fitness of each individual $\mathbf{C}_{g,w} \in C_g$
 - 5: $g = g + 1$
 - 6: Select parents from C_{g-1}
 - 7: Recombine selected parents through cross-over to form offspring O_g
 - 8: Mutate offspring in O_g
 - 9: Select the new generation C_g from the previous generation C_{g-1} and the offspring O_g
 - 10: **end while**
-

In the context of GAs, the fitness function is something like

$$F(\mathbf{C}) = a_1 F_1(\mathbf{C}) + a_2 F_2(\mathbf{C}) + \dots \quad (4)$$

i.e. a function which rates the chromosomes according to a number of criteria, the influence of each criterion being controlled by the related constant $a \in \mathbb{R}$. Function F represents a cost and returns lower values for better chromosomes. In general, also the a_i factors may be set as evolving parameters, adapting the weight of each F_i to the environment. This aspect is out of the scope of this paper and will be considered in future work.

3 An AEF-Based Overlay Scheme for Resource Sharing

In this section we introduce a GA-based AEF system which enables efficient and scalable sharing of consumable resources, *i.e.* resources that cannot be acquired (by replication) once discovered, but may only be directly used upon contracting with their hosts [17]. An example of consumable resource is disk space, which can be partitioned and allocated to requestors for the duration of a task, or in general for an arranged time. We would like to emphasize that this overlay scheme is not the main contribution of this paper. It is a simple AEF example, which can be easily interpreted as the evolutionary version of Gnutella [7], applied to consumable resource sharing rather than file sharing.

Algorithm 2 is executed by each peer involved in a resource discovery process, which is based on *epidemic* propagation of queries [6]. Query messages generated by a peer are matched with local resources and eventually propagated to neighbors. Each peer has a cache which contains advertisements of resources previously discovered in other peers. The cache is used as a preferred alternative to random (*i.e.* blind) query propagation. Each query message has a time-to-live (*TTL*) which is the remaining number of hops before the query message itself expires (*i.e.* it is no longer propagated). Moreover, each peer caches received queries, in order to drop subsequent duplicates. In general, bio-inspired epidemic protocols have considerable benefits as they are robust against network failures, scalable and provide probabilistic reliability guarantees [1].

Algorithm 2. Resource discovery

```

1: if resource locally available then
2:   Notify interested peer
3:   if interested peer still interested then
4:     Allocate resource for interested peer
5:     Add interested peer to neighbor list
6:   end if
7: else
8:   if resource advertisement in cache then
9:     Propagate query (TTL − 1) to cached peer
10:  else
11:    if TTL > 0 then
12:      Propagate query (TTL − 1) to some neighbors
13:    end if
14:  end if
15: end if

```

The resource discovery process is affected by the following parameters, representing the phenotype of each peer:

- f_k = fraction of neighbors targeted for query propagation
- TTL_{max} = max number of hops for messages
- D_{max} = max size of the cache

Traditional epidemic algorithms use fixed values for parameters, set in advance according to the results of some tuning process which should guarantee good performance with high probability. Actually, if all nodes would be configured with $f_k = 1$ and the same TTL value, the resulting scheme would be exactly Gnutella. In our scheme, parameters are functions of the chromosome, thus randomly initialized when a peer is created and joins the network. Moreover, adaptive tuning of parameters is based on GAs, with fitness function $F(\Phi, \langle QHR \rangle)$ to be minimized, where $\Phi = \Phi(f_k, TTL_{max}, D_{max})$ and

$$\langle QHR \rangle = \frac{1}{k+1} \sum_{j=0}^k QHR_j \quad (5)$$

QHR_j is the query hit ratio, *i.e.* the number of query hits QH versus the number of queries Q , assuming index $j = 0$ for current peer and $j = 1, \dots, k$ for its neighbors.

Being shared resources (*e.g.* CPU, RAM, disk space) consumable, the discovery process required by a job to start and complete its execution may be penalized by an increasing request rate. In general, the fitness function must be chosen in order to make each peer adapt to the rate of user requests which directly or indirectly affect its running environment.

If the values of the parameters which define the phenotype increase, the search process becomes more expensive in terms of time/space, but at the same time the discovery probability (which may be approximated by QHR) should result to be improved. In its most simple form

$$\Phi = \phi_0 f_k + \phi_1 TTL_{max} + \phi_2 D_{max} \quad (6)$$

where constant weights $\phi_0, \phi_1, \phi_2 \in \mathbb{R}$ control the influence of each parameter. In this case, the fitness function should reward peers with smaller Φ value, having the same *sufficiently high* $\langle QHR \rangle$ value.

The genotype \mathbf{C} of each peer is given by three genes $\{C_0, C_1, C_2\}$ with values in a limited subset of \mathbb{N} . The relationship between the phenotype and the genotype is given by the following equations:

- $f_k = c_0 C_0$
- $TTL_{max} = c_1 C_1$
- $D_{max} = c_2 C_2$

where $c_i \in \mathbb{R}$, (with $i = 0, 1, 2$) are constants.

The pseudocode in Algorithm 3 describes the adaptation process which is periodically executed by each peer.

The best neighbor is chosen using proportional selection, *i.e.* the chance of individuals (neighbors' chromosomes) of being selected is inversely proportional to their fitness values. Cross-over is always performed, with a randomly-generated crosspoint. Mutation depends on $\langle QHR \rangle$, being highly improbable while the average query hit ratio of the peer and its neighbors tends to 1. The final selection between current peer's chromosome and the mutated offspring is random, with probability that is inversely proportional to their fitness values.

Algorithm 3. REVOL adaptation scheme

```

1: Let  $g = 0$ 
2: while not converged do
3:   Evaluate the fitness of own chromosome
4:    $g = g + 1$ 
5:   Select neighbor with best fitting chromosome
6:   Perform cross-over with best neighbor to generate offspring  $O_g = \{O_{g1}, O_{g2}\}$ 
7:   Mutate offspring in  $O_g$  with probability  $1 - \langle QHR \rangle$ 
8:   Select the new generation  $C_g$  from the previous generation  $C_{g-1}$  and the offspring
    $O_g$ 
9: end while

```

We simulated this example scheme with the Discrete Event Universal Simulator (DEUS) [3]. This tool provides a simple Java API for the implementation of nodes, events and processes, and a straightforward but powerful XML schema for configuring simulations.

Among others, we considered a network with fixed size and nodes performing periodic adaptation based on the following fitness function:

$$F1(\Phi, \langle QHR \rangle) = \begin{cases} 1/\Phi & \text{if } \langle QHR \rangle < 0.99 \\ \Phi & \text{if } \langle QHR \rangle > 0.99 \end{cases}$$

The idea behind F1 is that when resource usage is too low (below the minimum needed to provide a good service, *i.e.* a good query hit ratio), the system adapts itself in order to increase it until the query hit ratio is optimum; after that, it tries to minimize resource usage.

If QHR decreases, the system reacts in a way that chromosomes with high gene values survive and proliferate. At the beginning of the simulation, all nodes have the default QHR value which is 0.5. After a while, almost all nodes have performed at least one search, thus they have a QHR value that reflects the results of their searches. As clearly illustrated by fig. 2, when QHR is low, the gene values tend to increase. This lead to successful searches, for which QHR increases and the gene values decrease to a stable low value.

4 Related Work

A constructive criticism to both epidemic algorithms and mechanisms based on spanning trees is provided by Merz and Gorunova [13]. Being epidemic algorithms easy to implement, fault-tolerant, scalable, but slow, and spanning trees much more efficient but unresilient, the authors propose an hybrid approach. Epidemic dissemination is used to maintain the overlay, with an efficient mechanism for multicasting information. Simulation results show that the combination of the two methods is more efficient than the methods alone. Scalability up to 10^5 peers is also shown. The authors defer to future work a detailed evaluation considering failures, high churn and presence of malicious peers.

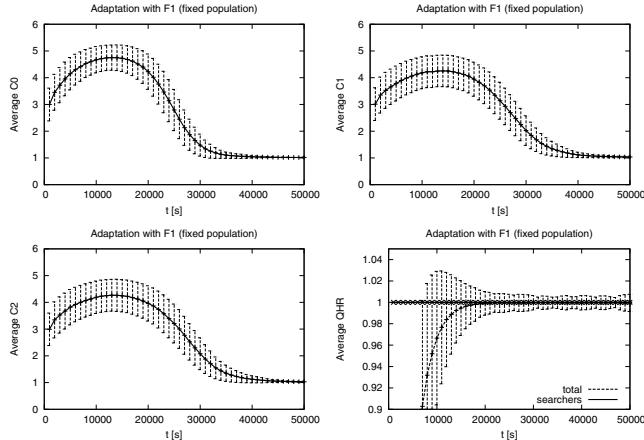


Fig. 2. Evolution over time of genes C_0, C_1, C_2 for a network with fixed population, fitness function F1, little load and adaptation taking place every 17 min

Kessler *et al.* [12] propose to use genetic algorithms to find optimal network structures, by means of offline simulations. Each individual population member (chromosome) encodes a network design with connected superpeers and clusters of leaf nodes forming superpeer groups. The fitness of a population member is directly related to how successful the particular network configuration is in answering resource queries. With respect to the GA-based AEF, for which evolution is an online process, this approach has many drawbacks.

A scheme that is suitable to be included in the AEF framework is illustrated in a recent work of Wickramasinghe *et al.* [18]. The authors describe and evaluate a fully distributed P2P evolutionary algorithm where decisions regarding survival and reproduction are taken by the individuals themselves independently, without any central control. This allows for a fully distributed evolutionary algorithm, where not only reproduction (crossover and mutation) but also selection is performed at local level.

5 Conclusion

In this paper we formalized the Adaptive Evolutionary Framework (AEF), which is a framework for the design of self-configuring peer-to-peer overlay schemes. To show the potential of AEF, we used it to define a resource sharing scheme in which the evolutionary aspect is driven by a genetic algorithm.

In prospect, we are interested in finding other possible AEF implementations than genetic algorithms, considering also more complicated environments and applications, for which adaptiveness is not a plus but a fundamental requirement.

References

1. Ahi, E., Caglar, M., Ozkasap, O.: Stepwise Fair-Share Buffering underneath Bio-inspired P2P Data Dissemination. In: 6th International Symposium on Parallel and Distributed Computing (ISPDC 2007), Hagenberg, Austria (2007)
2. Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2001)
3. Agosti, M., Amoretti, M.: Discrete Event Universal Simulator (DEUS), <http://code.google.com/p/deus/>
4. Eiben, A.E.: Evolutionary computing and autonomic computing: Shared problems, shared solutions? In: Babaoğlu, Ö., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., van Moorsel, A., van Steen, M. (eds.) SELF-STAR 2004. LNCS, vol. 3460, pp. 36–48. Springer, Heidelberg (2005)
5. Engelbrecht, A.: Computational Intelligence: An Introduction, 2nd edn. Wiley & Sons, Chichester (2007)
6. Eugster, P.T., Guerraoui, R., Kermarrec, A.-M., Massoulie, L.: From Epidemics to Distributed Computing. IEEE Computer 37(5) (2004)
7. Gnutella RFC homepage, <http://rfc-gnutella.sourceforge.net>
8. Hales, D.: From Selfish Nodes to Cooperative networks - Emergent Link-based incentives in Peer-to-Peer Networks. In: Proc. 4th IEEE Int'l. Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland (2004)
9. Heylighen, F.: The Science of Self-organization and Adaptivity. In: Kiel, L.D. (ed.) The Encyclopedia of Life Support Systems (EOLSS). Eolss Publishers, Oxford (2001)
10. Holland, J.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1992)
11. Kalyvianaki, E., Pratt, I.: Building Adaptive Peer-to-Peer Systems. In: 4th IEEE International Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland (2004)
12. Kessler, J., Rasheed, K., Budak Arpinar, I.: Using genetic algorithms to reorganize superpeer structure in peer to peer networks. Applied Intelligence 26(1), 35–52 (2007)
13. Merz, P., Gorunova, K.: Fault-tolerant Resource Discovery in Peer-to-peer Grids. Journal of Grid Computing 5(3) (2007)
14. Ottino, J.M.: Engineering complex systems. Nature 427, 399 (2004)
15. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001)
16. Schoder, D., Fischbach, K.: The Peer-to-Peer Paradigm. In: 38th Annual Hawaii International Conference on System Sciences (HICSS 2005), Hawaii, USA (2005)
17. Tang, J., Zhang, W., Xiao, W., Tang, D., Song, J.: Self-Organizing Service-Oriented Peer Communities. In: International Conference on Internet and Web Applications and Services (ICIW 2006), Guadeloupe, French Caribbean (2006)
18. Wickramasinghe, W., van Steen, M., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1460–1467. ACM Press, New York (2007)

Multiuser Scheduling in HSDPA with Particle Swarm Optimization*

Mehmet E. Aydin¹, Raymond Kwan¹, Cyril Leung², and Jie Zhang¹

¹ University of Bedfordshire, CWIND, Luton, UK

{mehmet.aydin,raymond.kwan,jie.zhang}@beds.ac.uk

² The University of British Columbia, Vancouver, B.C., Canada, V6T 1Z4
cleung@ece.ubc.ca

Abstract. In this paper, a mathematical model of multiuser scheduling problem in HSDPA is developed to use in optimization process. A more realistic imperfect channel state information (CSI) feedback, which is required for this problem, in the form of a finite set of Channel Quality Indicator (CQI) values is assumed, as specified in the HSDPA standard [1]. A global optimal approach and a particle swarm optimization approach are used to solve the problem. Simulation results indicate that the performances of the two approaches are very close even though the complexity of the particle swarm optimization approach is much lower.

1 Introduction

One of the effective ways for improving the spectral efficiency in a wireless communication system is the use of adaptive modulation and coding (AMC) [2,3]. A higher order modulation provides a better spectral efficiency at the expense of a worse error rate performance. A lower rate channel coding generally provides a better error rate performance at the cost of a poorer spectral efficiency. Thus, with a proper combination of the modulation order and channel coding rate, it is possible to design a set of modulation and coding schemes (MCS), from which a selection is made in an adaptive fashion in each transmission-time interval (TTI) so as to maximize system throughput under varying channel conditions. A common requirement is that the probability of erroneous decoding of a Transmission Block should not exceed some threshold value [1].

The downlink transmit power is often held constant [4] as in the HSDPA scheme while the above adaptation is based on the selection of the best MCS [1]. In addition, multiple orthogonal channelization codes (multicodes) can be used in transmitting data to a single user, thereby increasing the per-user bit rate and granularity of adaptation [1,5,6]. In Wideband Code-Division Multiple Access (WCDMA), such channelization codes are often referred to as Orthogonal

* This work is supported by EU FP6 "GAWIND" project on Grid-enabled Automatic Wireless Network Design under grant MTKD-CT-2006-042783 and EU FP7 Marie Curie International Incoming Fellowship METOP project under grant PIIF-GA-2008-221380.

Variable Spreading Factor (OVSF) codes; the number of OVSF codes per base station (BS) is limited due to their orthogonal property [5]. Thus, orthogonal code channels are also a scarce resource at the BS in addition to the downlink transmit power.

In exploiting multiuser diversity, a common way to increase the network throughput is to assign resources to a user with the largest signal-to-noise ratio (SNR) among all backlogged users (i.e. users with data to send) at the beginning of each scheduling period. However, due to limited mobile capability [4], a user might not be able to utilize all the radio resources available at the BS. Thus, transmission to multiple users during a scheduling period may be more resource efficient. In [7], the problem of downlink multiuser scheduling subject to limited code and power constraints is addressed. It is assumed in [7] that the exact path-loss and received interference power at every TTI for each user are fed back to the BS. This would require a large bandwidth overhead.

In this paper, the problem of optimal multiuser scheduling in HSDPA is addressed. The MCSs, numbers of multicodes and power levels for all users are jointly optimized at each scheduling period, given that only limited CSI information, as specified in the HSDPA standard [4], is fed back to the BS. An integer programming model is proposed in order to provide a globally optimal solution to the multiuser scheduling problem. Due to the complexity of the globally optimal method, a swarm intelligence approach, namely particle swarm optimization, is subsequently proposed. The experimentations suggest that it potentially provides a near-optimum performance with significantly reduced complexity.

2 System Domain

Let P_i be the default downlink transmit power to user i from a BS, h_i be the path gain between the BS and user i , and I_i be the total received interference and noise power at user i . The downlink received Signal to Interference Ratio (SIR) for user i is given by $\gamma_i = \frac{h_i P_i}{I_i}$, $i = 1, \dots, N$, subject to the power constraint

$$\sum_{i=1}^N P_i \leq P_T, \quad (1)$$

where N is the number of users, and P_T is the total power. Upon receiving the SIR value, γ_i , from user i , the BS decides on the most appropriate combination of MCS and number of multicodes for user i . If the continuous SIR value γ_i is fed back, an impractically large feedback channel bandwidth would be needed. In [4], the channel quality information fed back by a mobile, also known as the *channel quality indicator* (CQI), can only take on a finite number of non-negative integer values $\{0, 1, \dots, K\}$. The CQI is provided by the mobile via the High Speed Dedicated Physical Control Channel (HS-DPCCH). Each CQI value maps directly to a maximum bit rate¹ that a mobile can support, based on the

¹ In this paper, the bit rate refers to the transport block size, i.e. the maximum number of bits that a transport block can carry, divided by the duration of a TTI, i.e. 2 ms [5].

channel quality and mobile capability [8], while ensuring that the Block Error Rate (BLER) does not exceed 10%.

While the mapping between the CQI and the SIR is not specified in [4], this issue has been addressed in a number of proposals [9]-[11]. Recently, a mapping has been proposed in which the system throughput is maximized while the BLER constraint is relaxed [12]. Let $\tilde{\gamma}_i = 10 \log_{10}(\gamma_i)$ be the received SIR value at user i in dB and q_i be the CQI value that user i sends back to the BS via HS-DPCCH. According to [9,10,12], the mapping between q_i and $\tilde{\gamma}_i$ can generally be expressed as a piece-wise linear function

$$q_i = \begin{cases} 0 & \tilde{\gamma}_i \leq t_{i,0} \\ \lfloor c_{i,1}\tilde{\gamma}_i + c_{i,2} \rfloor & t_{i,0} < \tilde{\gamma}_i \leq t_{i,1} \\ q_{i,max} & \tilde{\gamma}_i > t_{i,1} \end{cases} \quad (2)$$

where $\{c_{i,1}, c_{i,2}, t_{i,0}, t_{i,1}\}$ are model and mobile capability dependent constants, and $\lfloor \cdot \rfloor$ is the floor function. From (2), it is clear that $\tilde{\gamma}_i$ cannot be recovered exactly based on the value of q_i alone due to the quantization. It is important to note that the region $t_{i,0} < \tilde{\gamma}_i \leq t_{i,1}$ is the operating region for the purpose of link adaptation and it should be large enough to accommodate SIR variations encountered in most practical scenarios [5]. In a well designed system, the probability that $\tilde{\gamma}_i$ lies outside this range should be small. As part of our proposed procedure, $\tilde{\gamma}_i$ can be approximated as

$$\tilde{\gamma}_i^\dagger = \tilde{\gamma}_i^{(l)} + (\tilde{\gamma}_i^{(u)} - \tilde{\gamma}_i^{(l)}) \xi, \quad (3)$$

where $\tilde{\gamma}_i^{(l)} = \frac{q_i - c_{i,2}}{c_{i,1}}$ and $\tilde{\gamma}_i^{(u)} = \frac{q_i + 1 - c_{i,2}}{c_{i,1}}$, and ξ follows a uniform distribution, i.e. $\xi \sim U(0, 1)$. Note that this approximation assumes that $\tilde{\gamma}_i$ is uniformly distributed between $\tilde{\gamma}_i^{(l)}$ and $\tilde{\gamma}_i^{(u)}$ for a given value of q_i .

When $q_i = 0$ and $q_i = q_{i,max}$, $\tilde{\gamma}_i$ can be simply approximated as $t_{i,0}$ and $t_{i,1}$ respectively, or more generally as $t_{i,0} - \xi_{i,0}$ and $t_{i,1} + \xi_{i,1}$ respectively, with $\xi_{i,0}$ and $\xi_{i,1}$ following certain pre-defined distributions. Finally, the estimated value of γ_i is given by $\hat{\gamma}_i = 10^{\tilde{\gamma}_i^\dagger/10}$. We refer to the mapping from SIR to q_i in (2) as the *forward mapping*, and the approximation of SIR based on the received value of q_i in (3) as the *reverse mapping*.

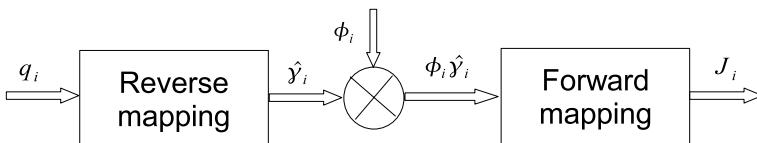


Fig. 1. The conversion process from the received CQI, q_i , from the mobile to the assigned rate index J_i at the base station

3 Multiuser Scheduling Problem

The CQI feedback value, q_i , from user i corresponds to the rate index that the user requests from the BS, and is associated with a required number of OVSF codes (multicodes) and downlink transmit power. Since the number of multicodes and transmit power are limited, the BS might not be able to simultaneously satisfy the bit rate requests for all users as described by $\{q_i, i = 1, \dots, N\}$. Thus, given the set $\{q_i, i = 1, \dots, N\}$, the BS must calculate a set of *modified CQIs*, $\{J_i, i = 1, \dots, N\}$, for all users by taking into account the power and number of multicodes constraints. By making use of the *forward* and *reverse* mappings in (2) and (3) respectively, the set of modified CQIs, $\{J_i, i = 1, \dots, N\}$, can be obtained as

$$J_i = \min \left(\max \left(\eta_i(\tilde{\gamma}_i^\dagger, \phi_i), 0 \right), q_{i,max} \right), \quad (4)$$

by assigning a power adjustment factor ϕ_i to user i , i.e. $\hat{\gamma}_i \mapsto \phi_i \hat{\gamma}_i$, where

$$\eta_i(\tilde{\gamma}_i^\dagger, \phi_i) = \lfloor c_{i,1} \left(\tilde{\gamma}_i^\dagger + 10 \log_{10} \phi_i \right) + c_{i,2} \rfloor, \quad (5)$$

$$0 \leq \phi_i \leq 10^{\left(\frac{q_{i,max} - (c_{i,1} \tilde{\gamma}_i^\dagger + c_{i,2})}{10 c_{i,1}} \right)}. \quad (6)$$

The conversion process from the received CQI, q_i , to the final assigned rate index, J_i , is sketched in Fig. 1. The optimal scheduling problem **P1** can be expressed as

$$\mathbf{P1} : \quad \max_{\mathbf{A}, \overline{\phi}} \sum_{i=1}^N \sum_{j=0}^{J_i} a_{i,j} r_{i,j} \quad (7)$$

subject to (6)

$$\sum_{j=0}^{J_i} a_{i,j} = 1, \quad \forall i, \quad (8)$$

$$\sum_{i=1}^N \sum_{j=0}^{J_i} a_{i,j} n_{i,j} \leq N_{max}, \quad (9)$$

$$\sum_{i=1}^N \phi_i \leq N, \quad (10)$$

where N_{max} is the maximum number of multicodes available for HSDPA at the BS, and $a_{i,j} \in \{0, 1\}$ is a binary optimization variable. The terms $r_{i,j}$ and $n_{i,j}$ correspond to the achievable bit rate and the required number of multicodes associated with CQI value j , $j \in \{0, 1, \dots, K\}$, for user i , and are given in [4]. Note that J_i is the maximum allowable CQI value for user i . Depending on code availability, the assigned combination of MCS and the number of multicodes may correspond to a bit rate that is smaller than that permitted by J_i . The

constraint in (10) can be obtained by substituting $P_i = \phi_i P_T / N$ into (1). The objective of the above optimization problem is to select the values of the decision variables $\mathbf{A} = \{a_{i,j}\}$ and $\bar{\phi} = \{\phi_i\}$ at each TTI in order to maximize (7), subject to (4)-(6) and (8)-(10).

4 Particle Swarm Optimization Algorithms (PSO)

PSO is one of the population based optimization technique inspired by social behavior of bird flocking and fish schooling. PSO inventors [13,14] were inspired by such scenarios based on natural processes explained below to solve the optimization problems. Suppose the following scenario: a group of birds are randomly searching for food in an area, where there is only one piece of food available and none of them knows where it is, but they can estimate how far it would be. The problem here is "what is the best strategy to find and get that food". Obviously, the simplest strategy is to follow the bird known as the nearest one to the food based on the estimation.

Analogous to this natural process, a population of individual solutions can be adopted as a flock/swarm, where each single solution, called particle, is considered as a bird/a member of the swarm. The ultimate aim is to explore for the piece of food/optimum solution within a particular area/search space. The achievement of each particle is measured with a fitness value calculated by a fitness function, and a velocity of flying towards the optimum, food. All particles fly across the problem space following the particle nearest to the optimum, food. PSO starts with initial population of solutions, which is evolved generation-by-generation towards the ultimate objective(s).

4.1 PSO Basics

The pure PSO algorithm builds each particle based on, mainly, two key vectors; position vector, $\mathbf{x}_i(\mathbf{t}) = \{x_{i,1}(t), \dots, x_{i,n}(t)\}$, and velocity vector $\mathbf{v}_i(\mathbf{t}) = \{v_{i,1}(t), \dots, v_{i,n}(t)\}$, where $x_{i,k}(t)$, is the position value of the i^{th} particle with respect to the k^{th} dimension ($k = 1, 2, 3, \dots, n$) at iteration t , and $v_{i,k}(t)$ is the velocity value of the i^{th} particle with respect to the k^{th} dimension at iteration t . The initial values, $\mathbf{x}_i(\mathbf{0})$ and $\mathbf{v}_i(\mathbf{0})$, are given by $x_{i,k}(0) = x_{min} + (x_{max} - x_{min}) \times r_1$, and $v_{i,k}(0) = v_{min} + (v_{max} - v_{min}) \times r_2$ where $x_{min}, x_{max}, v_{min}, v_{max}$ are lower and upper limits of the ranges of position and velocity values, respectively, and r_1 and r_2 are uniform random numbers in $[0, 1]$. Since both vectors are continuous, the original PSO algorithm can straightforwardly be used for continuous optimization problems. However, if the problem is combinatorial, a discrete version of PSO needs to be implemented. Once a solution is obtained, the quality of that solution is measured with a cost function denoted with f_i , where $f_i : \mathbf{x}_i(\mathbf{t}) \rightarrow \mathbb{R}$.

For each particle in the swarm, a personal best, $\mathbf{y}_i(\mathbf{t}) = \{y_{i,1}(t), \dots, y_{i,n}(t)\}$, is defined, where $y_{i,k}(t)$ denotes the position of the i^{th} personal best with respect to the k^{th} dimension at iteration t . The personal bests are equal to the corresponding initial position vector at the beginning. Then, in every generation, they are updated based on the solution quality. Regarding the objective function, f_i ,

the fitness values for the personal best of the i^{th} particle, $\mathbf{y}_i(\mathbf{t})$, is denoted by $f_i^y(t)$ and updated whenever $f_i^y(t+1) \prec f_i^y(t)$, where t stands for iteration and \prec corresponds to the logical operator, which becomes $<$ or $>$ for minimization or maximization problems respectively.

On the other hand, a global best, which is the best particle within the whole swarm is defined and selected among the personal bests, $\mathbf{y}(\mathbf{t})$, and denoted with $\mathbf{g}(\mathbf{t}) = \{g_1(t), \dots, g_n(t)\}$. The fitness of the global best, $f_g(t)$, can be obtained using

$$f_g(t) = \text{opt}_{i \in N} \{f_i^y(t)\} \quad (11)$$

where **opt** becomes min or max depending on the type of optimization. Afterwards, the velocity of each particle is updated based on its personal best, $\mathbf{y}_i(\mathbf{t})$ and the global best, $\mathbf{g}(\mathbf{t})$ using the following updating rule:

$$v_{i,k}(t+1) = \delta(w_t v_{i,k}(t) + c_1 r_1(y_{i,k}(t) - x_{i,k}(t)) + c_2 r_2(g_k(t) - x_{i,k}(t))) \quad (12)$$

where t is the time index, w_t is the inertia weight used to control the impact of the previous velocities on the current one. It is updated according to $w_{t+1} = \beta w_t$, where β is a decrement factor in $[0, 1]$. In (12), δ is a constriction factor which keeps the effects of the randomized weight within the certain range. In addition, r_1 and r_2 are random numbers in $[0, 1]$ and c_1 and c_2 are the learning factors, which are also called the social and cognitive parameters. Next, the positions are updated according to

$$x_{i,k}(t+1) = x_{i,k}(t) + v_{i,k}(t). \quad (13)$$

Subsequently, the fitness values corresponding to the updated positions are calculated, and the personal and global bests are determined accordingly. The whole process is repeated until a pre-defined stopping criterion is satisfied.

4.2 PSO Implementation

PSO has been applied in various continuous and discrete problems with good record of achievements [14,15,16]. This implementation is based on a standard PSO as described in subsection 4.1, but without the use of the velocity vector. The reason is due to the difficulty in its control in use, and its dispensability in computation. That means that, the use of velocity vector will oblige and require a two-phase adjustment process to keep the values within the range of the problem domain. This incurs more computational time, which is not desirable to solve such time-sensitive problems. Therefore, instead of (13), the rule

$$x_{i,k}(t+1) = x_{i,k}(t) + \Delta x_{i,k}(t) \quad (14)$$

is used, where t is the time index and $\Delta x_{i,k}(t)$ is calculated as

$$\Delta x_{i,k}(t) = \delta w_{t+1}(c_1 r_1(y_{i,k}(t) - x_{i,k}(t)) + c_2 r_2(g_k(t) - x_{i,k}(t))). \quad (15)$$

The mathematical model of the multiuser scheduling problem undertaken in this implementation is described in Section 3 with the constraints (4)-(10) and

the objective function (7). It is a maximization model to optimize two decision variables; $\bar{\phi}$ and \mathbf{A} , where $\bar{\phi}$ is a set of positive real numbers identifying the relative power level for each user, and \mathbf{A} is a set of binary variables which identifies the appropriate rate index, j , assigned to user, i , as described in section 3. Regarding these decision variables, the model is a typical mixed-integer programming model. The aim is to obtain the most appropriate values for both set of variables such that the throughput of the system is maximized as described in (7). In this paper, the problem is to explore the appropriate values of both $\bar{\phi}$ and \mathbf{A} using the technique of PSO.

The states of the problem are constructed based on $\bar{\phi}$ and \mathbf{A} , where the former is adopted as the main set of decision variables while the latter is to be fine-tuned eventually. For notational simplicity, let $\mathbf{a}_i = \{a_{i,0}, \dots, a_{i,J_i}\}$, $i = 1, \dots, N$. New solutions are generated by applying (14), which results in a particular value of ϕ_i , say, ϕ'_i . Correspondingly, \mathbf{a}_i is assigned to the highest available level, and a particular solution i can then be expressed as $\mathbf{x}_i = (\phi'_1, \dots, \phi'_i, \dots, \phi'_N, \mathbf{a}_1, \dots, \mathbf{a}_i, \dots, \mathbf{a}_N)$.

In other words, once ϕ'_i is generated, the corresponding J_i is calculated using (4), and a_{i,J_i} is assigned to be 1 while the rest of the elements are set to be 0,

i.e. $\mathbf{a}_i = \underbrace{\{0, 0, \dots, 1\}}_{J_i+1}$. If constraint (9) is violated, then the non-zero element of

$a_{i,j}$ is shifted backward by one position, i.e. $\mathbf{a}_i = \underbrace{\{0, 0, \dots, 1, 0\}}_{J_i+1}$. This process is repeated until (9) is satisfied, resulting an updated set of values \mathbf{a}'_i . Thus, the new solution can be expressed as $\mathbf{x}'_i = (\phi'_1, \dots, \phi'_i, \dots, \phi'_N, \mathbf{a}'_1, \dots, \mathbf{a}'_i, \dots, \mathbf{a}'_N)$.

Since PSO is a population based algorithm, in which the population evolves towards a predefined objective, (14) is applied to all particles and the whole swarm is updated as a result. In the next step, every particle within the swarm updates the personal best based on the new positions. Subsequently, the best of whole swarm is determined. As such, the algorithm carries out the search generation-by-generation.

The computational complexity of PSO can be discussed with regards to the number of users, N , which is the only variable that affects the problem size². Since the size of the swarm and the number of generations do not change instance by instance, the complexity corresponds to $\mathcal{O}(1)$. The remaining subject of the complexity is the number of users, N . The algorithm checks with all users to make sure that none of the constraints is violated. Therefore the complexity due to those checks is proportional to the size of the problem, which results in a complexity of $\mathcal{O}(N)$. In the worst case, the complexity of the algorithm is $\mathcal{O}(N)$.

5 Experimental Results

For illustration purposes, we have simulated different scenarios for evaluating the performance of the proposed algorithms. We first consider $N = 2$ users with

² In this paper, the number of available rates and the maximum number of codes are assumed fixed.

the parameter values in (2) as $t_{i,0} = -4.5$, $t_{i,1} = 25.5$, $c_{i,1} = 1$, $d_{i,1} = 4.5$, and $q_{i,max} = 30$ for $i = 1, 2$; these values are obtained from [9], assuming that the mobiles are of category 10 (i.e. have a wide CQI range) as defined in [4]. Values for $n_{i,j}$ and $r_{i,j}$ are obtained from [4]. Also, let $\{\gamma_i, i = 1, 2\}$ as defined in section 2 be the outcomes of Gamma distributed random variables $\{\Gamma_i, i = 1, 2\}$ with pdfs given by $f_{\Gamma_i}(\gamma) = \left(\frac{\alpha_i}{\bar{\Gamma}_i}\right)^{\alpha_i} \frac{\gamma^{\alpha_i-1}}{\Gamma(\alpha_i)} \exp\left(-\frac{\alpha_i \gamma}{\bar{\Gamma}_i}\right)$, $\gamma \geq 0$ [17], where $\Gamma(\cdot)$ is the Gamma function, α_i is the fading figure, and $\bar{\Gamma}_i$ is the mean of Γ_i . The Gamma distribution for the SIR is well-known in the area of wireless communications [17]. Let $\mathbf{\Gamma} = \{\Gamma_1 \ \ \Gamma_2\}$, and let the aggregate transport block size (TBS), T , per Transmission Time Interval (TTI) be $T = \sum_{i=1}^N \sum_{j=0}^{J_i} a_{i,j} r_{i,j}$.

We, first, solved the problem instances generated as 3 different scenarios with a global optimization algorithm, which we denote as Joint Global Optimum (JGO). Then, a simple greedy (SG) algorithm is used to draw a minimum level of solution quality. This SG simply allocates resources to the users in decreasing order of their estimated SIR values, $\hat{\gamma}_i$: the user with the highest $\hat{\gamma}$ is first allocated as many code resources as it can possibly use. Subsequent users are allocated in the same way until all code resources are exhausted. Finally, the same problem instances are solved using a PSO algorithm as described in section 4. Both the size of the swarm and the number of generations are kept as 100. The other PSO-related parameters are chosen to be $\delta = 1$, $w_0 = 0.98$, $\beta = 0.99998$ and $c_1 = c_2 = 1$. Parameters used in generating the benchmark scenarios are as follows: for Scenario I; ($\bar{\Gamma}_1 = 5, \alpha_1 = 6.5$), ($\bar{\Gamma}_2 = 6, \alpha_2 = 3$), for Scenario II; ($\bar{\Gamma}_1 = 12, \alpha_1 = 6.5$), ($\bar{\Gamma}_2 = 10, \alpha_2 = 3$), and for Scenario III; ($\bar{\Gamma}_1 = 17, \alpha_1 = 6.5$), ($\bar{\Gamma}_2 = 16, \alpha_2 = 3$).

The results are presented in Fig. 2 as the cumulative distribution function (CDF) of T for all three abovementioned different scenarios. As can be seen, the graphs show that the performances of PSO and JGO are very similar, and are consistently more superior than that of SG. The comparison of the average gains, i.e. $E_{\Gamma}[T]$, of PSO and JGO over SG for the three scenarios are as follows: for Scenario I, II, and III, PSO gains 18%, 6% and 17% while JGO gains 21%, 8% and 20%, respectively. These results show that a good throughput improvement can be achieved with JGO and PSO approaches over SG, and that the performances of JGO and PSO are very similar.

Fig. 3 shows the CDFs of T corresponding to the SG and PSO approaches for different number of users, over 2000 simulation instances. For each user, $\alpha=5$ and $\bar{\Gamma}=8.45$ dB. The CDF curves for JGO are not shown due to the excessively long running times but are expected to be close to those for PSO. The average performance comparison between different schemes for different user numbers of 2, 3, 4 and 5 are 11%, 12%, 12% and 15%, respectively. For each user, $\alpha=5$ and $\bar{\Gamma}=8.45$ dB. These summarize the comparison of $E_{\Gamma}[T]$ between PSO and SG approaches. It can be seen that the advantage of PSO over SG increases with the number of users. The results further justify the usefulness of the PSO approach.

The complexity of PSO is briefly discussed in Section 4.2, and is shown to be on the order of $\mathcal{O}(N)$. Table 1 shows the normalized CPU time, i.e. the CPU time relative to that of the 2-user case, for all three algorithms, based on 25

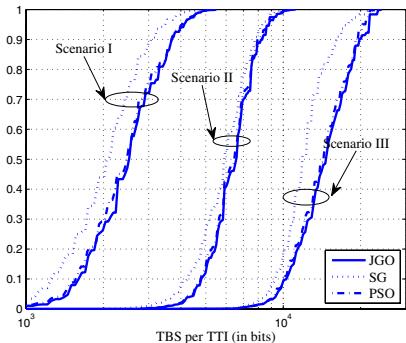


Fig. 2. CDF of the total number of multicodes at each TTI for 2 user cases

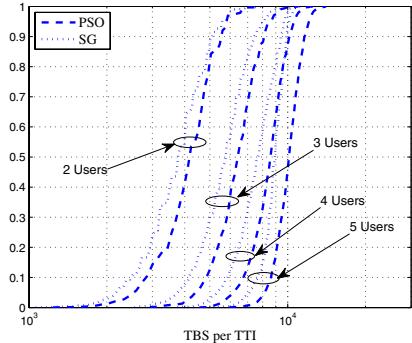


Fig. 3. CDF of the total number of multicodes at each TTI for 2, 3, 4, and 5 user cases

Table 1. Normalized CPU time for all three approaches on growth of problem size

# of Users	2	3	4	5
SG	1.00	1.32	1.76	2.11
JGO	1.00	5.10	49.80	961.09
PSO	1.00	1.33	1.67	2.00

simulation instances. The idea is to reveal the trend behind the computational complexity of the algorithms. It can be seen that the growth in complexity for JGO appears to be of an exponential nature, while those for the other two algorithms are linear, and are very similar. Such an observation agrees with the complexity analysis for PSO as presented in Section 4.2.

6 Conclusion

Two methods for allocating resources to multiple users in the context of HSDPA were proposed in conjunction with a problem formulation which adopts the channel feedback scheme specified in the WCDMA standard. Simulation results suggest that the proposed optimization methods can provide a substantial throughput improvement over a greedy approach. The performance of the PSO algorithm is found to be very close to that of the optimal algorithm, but with a linear complexity. The advantage of the proposed methods increases with the number of users.

References

1. Holma, H., Toskala, A. (eds.): HSDPA/HSUPA For UMTS High Speed Radio Access for Mobile Communications. John Wiley & Sons, Chichester (2006)
2. Alouini, M.S., Goldsmith, A.J.: Adaptive Modulation over Nakagami Fading Channels. In: Wireless Personal Communications, vol. 13, pp. 119–143. Kluwer Academic Publishers, Netherlands (2000)

3. Falahati, S.: Adaptive Modulation systems for Predicted Wireless Channels. *IEEE Transactions on Communications* 52(2), 307–316 (2004)
4. Universal Mobile Telecommunications Systems (UMTS); Physical Layer Procedures (FDD). Technical Specification 3GPP TS25.214, 3rd Generation Partnership Project (2007)
5. Holma, H., Toskala, A. (eds.): WCDMA for UMTS Radio Access for Third Generation Mobile Communications, 3rd edn. John Wiley & Sons, Chichester (2004)
6. Lee, S.J., Lee, H.W., Sung, D.K.: Capacities of Single-Code and Multicode DS-CDMA Systems Accommodating Multiclass Service. *IEEE Trans. on Vehicular Technology* 48(2), 376–384 (1999)
7. Kwan, R., Leung, C.: Downlink Scheduling Schemes for CDMA Networks with Adaptive Modulation and Coding and Multicodes. *IEEE Transactions on Wireless Communications* 6(10), 3668–3677 (2007)
8. Universal Mobile Telecommunications Systems (UMTS); UE Radio Access Capabilities. Technical Specification 3GPP TS25.306, 3rd Generation Partnership Project (2007)
9. Motorola, Nokia: Revised CQI Proposal. Technical Report R1-02-0675, 3GPP RAN WG1 (April 2002)
10. Brouwer, F., de Bruin, I., Silva, J.C., Souto, N., Cercas, F., Correia, A.: Usage of Link-Level Performance Indicators for HSDPA Network-Level Simulation in E-UMTS. In: Proc. of International Symposium on Spread Spectrum Techniques and Applications (ISSSTA), Sydney, Australia (September 2004)
11. Ko, K., Lee, D., Lee, M., Lee, H.: Novel SIR to Channel-Quality Indicator (CQI) mapping method for HSDPA System. In: Proc. of IEEE Vehicular Technology Conference (VTC), Montreal, Canada (September 2006)
12. Freudenthaler, K., Springer, A., Wehinger, J.: Novel SINR-to-CQI Mapping Maximizing the Throughput in HSDPA. In: Proc. of IEEE Wireless Communications and Networking Conference (WCNC), Hong Kong, China (March 2007)
13. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of the 6th Int. Symposium on Micro-Machine and Human Science, 1995, pp. 39–43 (1995)
14. Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann, San Mateo (2001)
15. Salman, A., Ahmad, I., Al-Madani, S.: Particle Swarm Optimization for Task Assignment Problem. *Microprocessors and Microsystems* 26, 363–371 (2003)
16. van den Bergh, F., Engelbecht, A.: Cooperative Learning in Neural Networks Using Particle Swarm Optimizers. *South African Computer Journal* 26, 84–90 (2000)
17. Simon, M.K., Alouini, M.S.: Digital Communication over Fading Channels, 2nd edn. John Wiley & Sons, Chichester (2005)

Efficient Signal Processing and Anomaly Detection in Wireless Sensor Networks*

Markus Wälchli and Torsten Braun

Institute of Computer Science and Applied Mathematics
University of Bern, Switzerland
waelchli@iam.unibe.ch

Abstract. In this paper the node-level decision unit of a self-learning anomaly detection mechanism for office monitoring with wireless sensor nodes is presented. The node-level decision unit is based on Adaptive Resonance Theory (ART), which is a simple kind of neural networks. The Fuzzy ART neural network used in this work is an ART neural network that accepts analog inputs. A Fuzzy ART neural network represents an adaptive memory that can store a predefined number of prototypes. Any observed input is compared and classified in respect to a maximum number of M online learned prototypes. Considering M prototypes and an input vector size of N , the algorithmic complexity, both in time and memory, is in the order of $O(MN)$. The presented Fuzzy ART neural network is used to process, classify and compress time series of event observations on sensor node level. The mechanism is lightweight and efficient. Based on simple computations, each node is able to report locally suspicious behavior. A system-wide decision is subsequently performed at a base station.

Keywords: Sensor networks, anomaly detection, pattern recognition.

1 Introduction

Wireless sensor networks have a number of strengths such as distributivity, parallelism, redundancy, and comparatively high cost-effectiveness due to lack of wires. On the other hand, their tininess, need for long-term operation, and dependency on batteries impose severe restrictions on the system. Hence, services provided in sensor networks need to be lightweight in terms of memory and processing power and should not require high communication costs.

The goal of this work is to provide an office monitoring system which is able to distinguish abnormal office access from normal access. Therefore, office access patterns need to be classified. The access patterns vary in their time of presence. They are composed of signals, i.e., collected time series of observations of some

* The work presented in this paper was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the SNF under grant number 5005-67322.

phenomena, which are monitored on the sensor nodes. This classification problem faces mainly two restrictions in sensor networks. First, processing power and memory are limited on sensor nodes. Accordingly, complex pattern classification methods such as presented in [1] are difficult to be implemented on node level. Second, communication costs are high. Therefore, it is not possible to transmit the observed signals without further processing to a fusion center. A possible solution to the problem could be based on querying systems [2]. However, those events that have to be monitored need to be defined and declared by a system expert. The approaches require a priori knowledge about the occurring events and thresholds to determine the event boundaries, which prevents any dynamic anomaly detection. Context-aware and Time Delayed Neural Networks [3] have been applied to classify events that evolve over time. However, they need periodic learning and specific neural networks for every kind of occurring event. Apart from classification, anomaly detection has been gained some attention too [4], [5]. The focus of these approaches is very specific and comparatively high communication and computation costs are accepted. Anomaly detection has furthermore been addressed by Artificial Immune Systems (AIS) [6], [7]. These systems work similar as ART neural networks, but are less compact and require more memory.

In our work, signals monitored over predefined intervals are classified on sensor nodes by an instance-based learning algorithm, where prototypic event patterns are dynamically learned, and infrequently matched patterns can be replaced. This algorithm is comparatively lightweight and accounts for the constraints in wireless sensor networks. Both learning and classification are performed by a Fuzzy ART neural network. Such kinds of networks have been implemented for sensor networks [8], [5]. However, the current focus has been on merging multiple sensor readings at discrete time points, rather than processing time series of measurements in an efficient and accurate manner.

To solve the classification problem we propose a two-layered approach. The observed signals are periodically collected and classified on the sensor nodes. The classifications assign the signals to learned prototypes. These assignments (classification numbers) are then periodically sent to a fusion center where the classification of the system-wide access pattern is performed. Accordingly, on each sensor node a high compression level is achieved. The Fuzzy ART neural network is tailored to the specific requirements of wireless sensor networks.

2 Classification and Anomaly Detection

Adaptive Resonance Theory (ART) [9] is a special kind of neural network with sequential learning ability. By feeding new samples, an ART neural network is able to refresh an already known prototype if the sample is similar to it. Else, a new category is created unless the total memory capacity has been utilized. Initially, ART networks were developed for binary input patterns. However, in subsequent work (Fuzzy ART) the network architecture had been enhanced to accept analog input samples too. The principle architecture is shown in Fig. 1.

A Fuzzy ART system operates unsupervised. It consists of two layers, a comparison layer F_1 with N neurons and a recognition layer F_2 composed of M

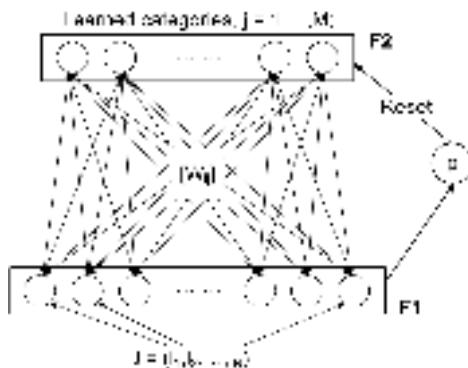


Fig. 1. Fuzzy ART neural network architecture

neurons. Neurons in F1 represent the attributes of the input, while neurons in F2 represent categories (prototypes). Furthermore, there is a sensitivity threshold ρ , also called vigilance factor, which evaluates the similarity between a given input I and any of the learned categories in F2. Similarity is always determined according to the Euclidean distance between I and the current category. The vigilance factor has impact on the system behavior: the higher the vigilance factor is chosen, the more fine-grained is the memory. This requires many categories, though. A last parameter that has impact on the system behavior is the learning rate. The higher the learning rate, the higher is the impact of the current input.

Fuzzy ART-based Learning

```

input: Input vector  $I$ ;
output: Number representing category  $j$  to which  $I$  belongs;
begin
    Compute the activation  $\alpha_j$  of each neuron in F2;
    Sort the  $\alpha_j$  in descending order;
    for each  $\alpha_j$  do
        Compute similarity  $s_j$  between  $I$  and category  $j$ ;
        if  $s_j > \rho$ 
            Update the weights in  $W_{i,j}$ ;
            return Category number  $j$ ;
    end
    if maximum number of categories is not reached
        Commit uncommitted neuron  $n$  in F2;
        return  $n$ ;
    else return -1;
end

```

The operation of a Fuzzy ART neural network is described in the pseudo-code above. Similarity is computed according to the weights stored in the matrix $W_{i,j}$. These weights represent the long-term memory of the system. The input vector I is compared to all categories. If I is similar to one of the stored categories (prototypes) the category number is returned as classification output and the weight matrix is updated. Else, either a new category is learned, i.e., if some memory in F2 is available, or the system returns -1.

Sensor networks have to face two main restrictions: Communication costs and memory constraints. On the other hand, anomaly detection requires the analysis of time series of signals. Fuzzy ART systems are ideally suited to meet these requirements: The input is sequentially processed, no buffering is required and time and storage complexity of a Fuzzy ART neural network are only in the order of $O(MN)$ [10], where M is the number of categories in F2 and N is the input vector size. The output of a Fuzzy ART neural network is the classification number of the input vector \mathbf{I} . Thus, a data compression of N to 1 is possible.

The sampling frequency is chosen depending on the observed signal. In the current implementation light is sampled 40 times in two seconds. The non-preprocessed input vector \mathbf{I} has a size of 40 elements. To decrease this complexity, two discrete Haar Wavelet transforms [11] are applied on the raw time series. Thus, the size of \mathbf{I} is reduced to 10. In addition to the needed reduction in sample size, the Wavelet transform also smooths the original input signal, which can either be interpreted as a smoothing of the original signal or as a generalization of the same. The discrete Haar Wavelet transform is applied as a simple digital Low Pass Filter (LPF), which can easily be performed on a sensor node. A data reduction factor of two for each application of the LPF is achieved.

With an unmodified Fuzzy ART network, non-classifiable signals evolve as soon as the memory is full. Any new pattern, even if it occurs frequently, could not be classified afterwards. This might make any evaluation difficult. Therefore, we support expiration. Based on an aging mechanism, always the oldest prototype is replaced with the latest one, else not classifiable, input sample. This approach is reasonable as frequently matched categories (normal behavior) will hardly be affected by the aging mechanism. Thus, anomalous behavior can be detected, even though infrequently used knowledge is forgotten by the system.

3 Signal Processing and Classification Performance

The performance of the Fuzzy ART neural network for anomaly detection and compression on node-level has been evaluated. TmoteSky [12] sensor nodes have been used. They consist of a microprocessor, 10 kB RAM, an IEEE 802.15.4 compliant radio, and a few sensors, whereof the light sensor has been used. Light is measured 40 times in an interval of 2 s. Two discrete Haar Wavelet transforms are applied. The resulting input vector \mathbf{I} of size 10 is fed into the Fuzzy ART neural network. The comparison layer F2 allocates memory for 10 categories, leading to storage requirements in the order of 200 bytes for the Fuzzy ART network. The vigilance factor ρ is 0.75 and the learning rate is 0.1.

The goal of the evaluation is to detect and report flashlight periods, which impose frequent switches from dark to bright and vice versa, in a dark room. Two different patterns of light activation have been performed. Both started during daytime, lasted for 10 minutes (see Fig. 2). Each pattern has been repeated three times. The sensor node was placed in the middle of the room.

Two representative runs are shown in Fig. 3. The other runs are similar. Instead of the classification numbers, only state changes, i.e., category switches

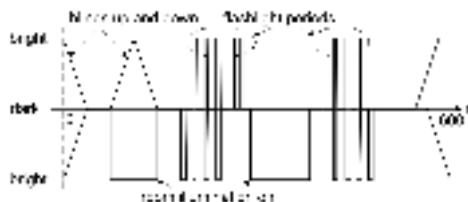


Fig. 2. Light pattern I (upper part) and light pattern II (lower part)

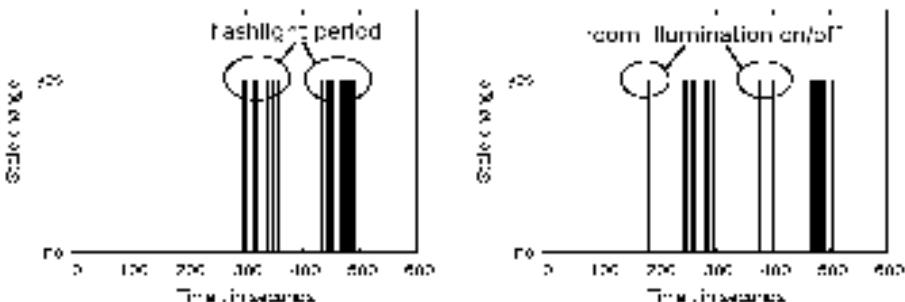


Fig. 3. Temporal state changes of the Fuzzy ART network

in the output stream of the Fuzzy ART network, are shown. Monitoring state changes is sufficient to distinguish exceptions from normal behavior. Thus the detection problem can be reduced to an analysis of sequences of binary decisions.

In all experiments the lowering and raising of the sun-blinds did not lead to any state changes. Accordingly, the Fuzzy ART anomaly detector is able to adapt to slowly changing environmental conditions. Both flashlight periods introduced many different input signals I , which resulted in many different classifications and accordingly in many state changes. The experiments with light pattern II contain two periods where the room illumination has turned on and off. This invokes abrupt illumination changes which are well observable by the few state changes before, respectively after the first flashlight period (on the left in Fig. 3). The average number of state changes in all twelve flashlight periods is 24.5 with a standard deviation of 2. Thus, a person moving with a flashlight could be easily detected by observing a certain number of state changes in a given interval.

4 Conclusions and Future Work

In this paper the compression and classification of anomalous behavior with a Fuzzy ART neural network on sensor node level has been addressed. Any observed time series (access pattern) fed to the system is mapped to a single classification value, which is sent to a fusion center. The Fuzzy ART neural network is self-learning, processes any input sequentially, needs no buffering

of samples, and adapts to both, changing environmental conditions and new evolving signals. Finally, the high compression rate lowers communications costs. In future work the fusion of local anomaly reports on a dedicated node will be considered. The goal is a working anomaly detection system for office monitoring.

References

1. Li, D., Wong, K.D., Hu, Y.H., Sayeed, A.M.: Detection, classification and tracking of targets. *IEEE Signal Processing Magazine* 19(2), 17–29 (2002)
2. Römer, K.: Discovery of frequent distributed event patterns in sensor networks. In: Verdone, R. (ed.) *EWSN 2008*. LNCS, vol. 4913, pp. 106–124. Springer, Heidelberg (2008)
3. Cai, J., Ee, D., Pham, B., Roe, P., Zhang, J.: Sensor network for the monitoring of ecosystem: Bird species recognition. In: Proc. of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP 2007), Melbourne, Australia, pp. 293–298 (2007)
4. Wang, X.R., Lizier, J.T., Obst, O., Prokopenko, M., Wang, P.: Spatiotemporal anomaly detection in gas monitoring sensor networks. In: Verdone, R. (ed.) *EWSN 2008*. LNCS, vol. 4913, pp. 90–105. Springer, Heidelberg (2008)
5. Li, Y.Y., Parker, L.: Intruder detection using a wireless sensor network with an intelligent mobile robot response. In: Proc. of the IEEE Southeast Con. 2008, Huntsville, Alabama, USA, pp. 37–42 (2008)
6. Dasgupta, D., Forrest, S.: Novelty detection in time series data using ideas from immunology. In: Proc. of The Fifth International Conference on Intelligent Systems (IS 1996), Reno, Nevada, USA (1996)
7. Mazhar, N., Farooq, M.: BeeAIS: Artificial Immune System Security for Nature Inspired, MANET Routing Protocol, BeeAdHoc. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) *ICARIS 2007*. LNCS, vol. 4628, pp. 370–381. Springer, Heidelberg (2007)
8. Kulakov, A., Davcev, D.: Intelligent wireless sensor networks using fuzzyart neural-networks (iscc 2007). In: Proc. of the 12th IEEE Symposium on Computers and Communications, Aveiro, Portugal, pp. 569–574. IEEE, Los Alamitos (2007)
9. Carpenter, G., Grossberg, S.: Adaptive Resonance Theory. Bradford Books. MIT Press, Cambridge (2002)
10. Burwick, T., Joublin, F.: Optimal algorithmic complexity of fuzzy art. *Neural Processing Letters* 7(1), 37–41 (1998)
11. Haar, A.: Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen* 69, 331–371 (1910)
12. Sentilla: Pervasive computing solutions (July 2008), <http://www.sentilla.com/>

Peer-to-Peer Optimization in Large Unreliable Networks with Branch-and-Bound and Particle Swarms^{*,**}

Balázs Bánhelyi¹, Marco Biazzini², Alberto Montresor², and Márk Jelasity³

¹ University of Szeged, Hungary

banhelyi@inf.u-szeged.hu

² University of Trento, Italy

{biazzini, montresor}@dit.unitn.it

³ University of Szeged and HAS, Hungary

jelasity@inf.u-szeged.hu

Abstract. Decentralized peer-to-peer (P2P) networks (lacking a GRID-style resource management and scheduling infrastructure) are an increasingly important computing platform. So far, little is known about the scaling and reliability of optimization algorithms in P2P environments. In this paper we present empirical results comparing two P2P algorithms for real-valued search spaces in large-scale and unreliable networks. Some interesting, and perhaps counter-intuitive findings are presented: for example, failures in the network can in fact significantly *improve* performance under some conditions. The two algorithms that are compared are a known distributed particle swarm optimization (PSO) algorithm and a novel P2P branch-and-bound (B&B) algorithm based on interval arithmetic. Although our B&B algorithm is not a black-box heuristic, the PSO algorithm is competitive in certain cases, in particular, in larger networks. Comparing two rather different paradigms for solving the same problem gives a better characterization of the limits and possibilities of optimization in P2P networks.

1 Introduction

Whereas large scale parallel computing is normally performed using GRID technologies, it is an emerging area of research to apply peer-to-peer algorithms in distributed global optimization. P2P algorithms can replace some of the centralized mechanisms of GRIDs that include monitoring and control functions. For example, network nodes can distribute information via “gossiping” with each other and they can collectively compute aggregates of distributed data (average, variance, count, etc) to be used to guide the search process [1]. This in turn increases robustness and communication efficiency, allows for a more fine-grained control over the parallel optimization process, and makes

* Full length version at <http://eprints.biblio.unitn.it/archive/00001541/>.

This work was supported by the European Space Agency through Ariadna Project “Gossip-based strategies in global optimization” (21257/07/NL/CB). M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. B. Bánhelyi was supported by the Ferenc Deák Scholarship No. DFÖ 19/2007, Aktion Österreich-Ungarn 70öu1, and OTKA T 048377.

** This work was partially supported by the Future & Emerging Technologies unit of the European Commission through Project CASCADAS (IST-027807).

it possible to utilize large-scale resources without a full GRID control layer and without reliable central servers.

The interacting effects of problem difficulty, network size, and failure patterns on optimization performance and scaling behavior are still poorly understood in P2P global optimization. In this paper we present empirical results comparing two P2P algorithms for real-valued search spaces in large-scale and unreliable networks: a distributed particle swarm optimization (PSO) algorithm [2] and a novel P2P branch-and-bound (B&B) algorithm based on interval arithmetic. Although our B&B algorithm is not a black-box heuristic, the PSO algorithm is competitive in certain cases, in particular, in larger networks. Some interesting, and perhaps counter-intuitive findings are presented as well: for example, failures in the network can in fact significantly *improve* the performance of P2P PSO under some conditions.

Other P2P heuristic optimization algorithms include [3,4,2]. They all build on gossip-based techniques [1,5] to spread and process information, as well as to implement algorithmic components such as selection and population size control. Our focus is somewhat different from [3,4] where it was assumed that population size is a fixed parameter that needs to be maintained in a distributed way. Instead, we assume that the network size is given, and should be exploited as fully as possible to optimize speed. In this context we are interested in understanding the effect of network size on performance, typical patterns of behavior, and related scaling issues.

In the case of B&B, we are not aware of any fully P2P implementations. The closest approach is [6], where some components, such as broadcasting the upper bound, are indeed fully distributed, however, some key centralized aspects of control remain, such as the branching step and the distribution of work. In unreliable environments we focus on, this poses a critical problem for robustness.

2 The Algorithms

Our target networking environment consists of independent nodes that are connected via an error-free message passing service: each node can pass a message to any target node, provided the address of the target node is known. We assume that node failures are possible. Nodes can leave and new nodes can join the network at any time as well.

In our P2P algorithms all nodes have identical roles running identical algorithms. Joining and failing nodes are tolerated automatically via the inherent (or explicit) redundancy of the algorithm design. This is made possible via a randomized communication substrate, the *peer sampling service* both algorithms are based on.

The peer sampling service enables all the nodes to send a message to a *random* node from the network at any time. This very simple communication primitive is called the *peer sampling service* that has a wide range of applications [7]. In this paper we will use this service as an abstraction, without referring to its implementation; lightweight, robust, fully distributed implementations exist based on the analogy of *gossip* [7]. The algorithms below will rely on two particular applications of the peer sampling service. The first is gossip-based broadcasting where, nodes periodically communicate pieces of information they consider “interesting” to random other nodes. This way, information spreads exponentially fast. The second is diffusion-based load balancing, where nodes

Algorithm 1. P2P B&B

```

1: loop                                ▷ main loop
2:    $I \leftarrow \text{priorityQ.getFirst}()$       ▷ most promising interval; if queue empty, blocks
3:    $(I_1, I_2) \leftarrow \text{branch}(I)$       ▷ cut the interval in two along longest side
4:    $\min_1 \leftarrow \text{upperBound}(I_1)$       ▷ minimum of 8 random samples from interval
5:    $\min_2 \leftarrow \text{upperBound}(I_2)$ 
6:    $\min \leftarrow \min(\min, \min_1, \min_2)$       ▷ current best value known locally
7:    $b_1 \leftarrow \text{lowerBound}(I_1)$           ▷ calculates bound using interval arithmetic
8:    $b_2 \leftarrow \text{lowerBound}(I_2)$ 
9:    $\text{priorityQ.add}(I_1, b_1)$             ▷ queue is ordered based on lower bound
10:   $\text{priorityQ.add}(I_2, b_2)$ 
11:   $\text{priorityQ.prune}(\min)$            ▷ remove entries with a higher lower bound than min
12:   $p \leftarrow \text{getRandomPeer}()$         ▷ calls the peer sampling service
13:   $\text{sendMin}(p, \min)$                   ▷ gossips current minimum
14:  if  $p$  has empty queue or local second best interval is better than  $p$ 's best then    ▷ gossip-based load balancing step
15:     $\text{sendInterval}(p, \text{priorityQ.removeSecond}())$ 
16:  procedure ONRECEIVEINTERVAL( $I (\subseteq D)$ ,  $b$ )                                ▷  $D \subseteq \mathbb{R}^d$  is the search space,  $b$  is lower bound of  $I$ 
17:     $\text{priorityQ.add}(I, b)$ 
18:  procedure ONRECEIVEMIN( $\min_p$ )
19:     $\min \leftarrow \min(\min_p, \min)$ 

```

periodically test random other nodes to see whether those have more load or less load, and then perform a balancing step accordingly. This process models the diffusion of the load over a random network.

Based on gossip-based broadcasting, a distributed implementation of a PSO algorithm was proposed [2]. Here we will use a special case of this algorithm, where particles are mapped to nodes: one particle per node. The current best solution, a key guiding information in PSO, is spread using gossip-based broadcast. This means we have a standard PSO algorithm where the number of particles equals the network size and where the neighborhood structure is a dynamically changing random network.

We now move on to the discussion of P2P B&B. Various parallel implementations of the B&B paradigm are well-known [8]. Our approach is closest to the work presented in [9] where the bounding technique is based on interval-arithmetic [10]. The important differences stem from the fact that our approach is targeted at the P2P network model described above, and it is based on gossip instead of shared memory. The basic idea is that, instead of storing it in shared memory, the lowest known upper bound of the global minimum is broadcast using gossip. In addition, the intervals to be processed are distributed over the network using gossip-based load balancing. The algorithm that is run at all nodes is shown in Algorithm 1. The lower bound for an interval is calculated using interval arithmetic [10], which guarantees that the calculated bound is indeed a lower bound. We start the algorithm by sending the search domain D with lower bound $b = \infty$ to a random node.

3 Experimental Results

The algorithms described above were compared empirically using the P2P network simulator PeerSim [11]. We selected well-known test functions as shown in Table 1. We included Sphere2 and Sphere10 as easy unimodal functions. Griewank10 is similar to Sphere10 with high frequency sinusoidal “bumps” superimposed on it. Schaffer10 is a sphere-symmetric function where the global minimum is surrounded by deceptive

Table 1. Test functions. D : search space; $f(x^*)$: global minimum value; K : # local minima.

	Function $f(x)$	D	$f(x^*)$	K
Sphere2	$x_1^2 + x_2^2$	$[-5.12, 5.12]^2$	0	1
Sphere10	$\sum_{i=1}^{10} x_i^2$	$[-5.12, 5.12]^{10}$	0	1
Griewank10	$\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^{10}$	0	$\approx 10^{19}$
Schaffer10	$0.5 + (\sin^2(\sqrt{\sum_{i=1}^{10} x_i^2}) - 0.5)/((1 + (\sum_{i=1}^{10} x_i^2)/1000)^2)$	$[-100, 100]^{10}$	0	≈ 63 spheres
Levy4	$\sin^2(3\pi x_1) + \sum_{i=1}^3 (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_4 - 1)(1 + \sin^2(2\pi x_4))$	$[-10, 10]^4$	-21.502356	71000

spheres. These two functions were designed to mislead local optimizers. Finally, Levy4 is not unlike Griewank10, but more asymmetric, and involves higher amplitude noise as well. Levy4 is in fact specifically designed to be difficult for interval arithmetic-based approaches.

We considered the following parameters, and examined their interconnection during the experiments: **network size** (N): the number of nodes in the network; **running time** (t): the duration while the network is running (note that it is not the sum of the running time of the nodes), the unit of time is one function evaluation; **function evaluations** (E): the number of overall function evaluations performed in the network; **quality** (ϵ): the difference of the fitness of the best solution found in the entire network and the optimal fitness. For example, if $t = 10$ and $N = 10$ then we know that $E = 100$ evaluations are performed.

Messages are delayed by a uniform random delay drawn from $[0, t_{eval}/2]$ where t_{eval} is the time for one function evaluation. In fact, t_{eval} is considerable in realistic problems, so our model of message delay is rather pessimistic. To simulate churn, in some of the experiments 1% of nodes are replaced during a time interval taken by 20 function evaluations. The actual wall-clock time of one function evaluation has a large effect on how realistic this setting is. We assume that the startup of the protocol is synchronous, that is, all nodes in the network are informed at a certain point in time that the optimization process should begin.

We focus on two key properties: (i) scaling with the constraint of a fixed amount of available function evaluations, and (ii) with the constraint of having to reach a certain solution quality. Our first set of experiments involves running the two algorithms with and without churn until 2^{20} function evaluations are consumed.¹

Solution quality is illustrated in Figure 1. Due to severe length restrictions we comment on the plots very briefly: we can see that (1) P2P B&B has the good property of refusing to utilize all resources if the function is too easy, eg Sphere; (2) B&B can never benefit from larger network sizes w.r.t. quality by definition, whereas PSO can; (3) on Levy4 PSO outperforms the more “sophisticated” B&B; (4) churn is always harmful for B&B by definition whereas it can be beneficial for PSO!

¹ We note here that for B&B, one cycle of Algorithm 1 was considered to take 20 evaluations, that is, in addition to the $2 \cdot 8 = 16$ normal evaluations, the interval-evaluation was considered to be equivalent to 4 evaluations (based on empirical tests on our test functions).

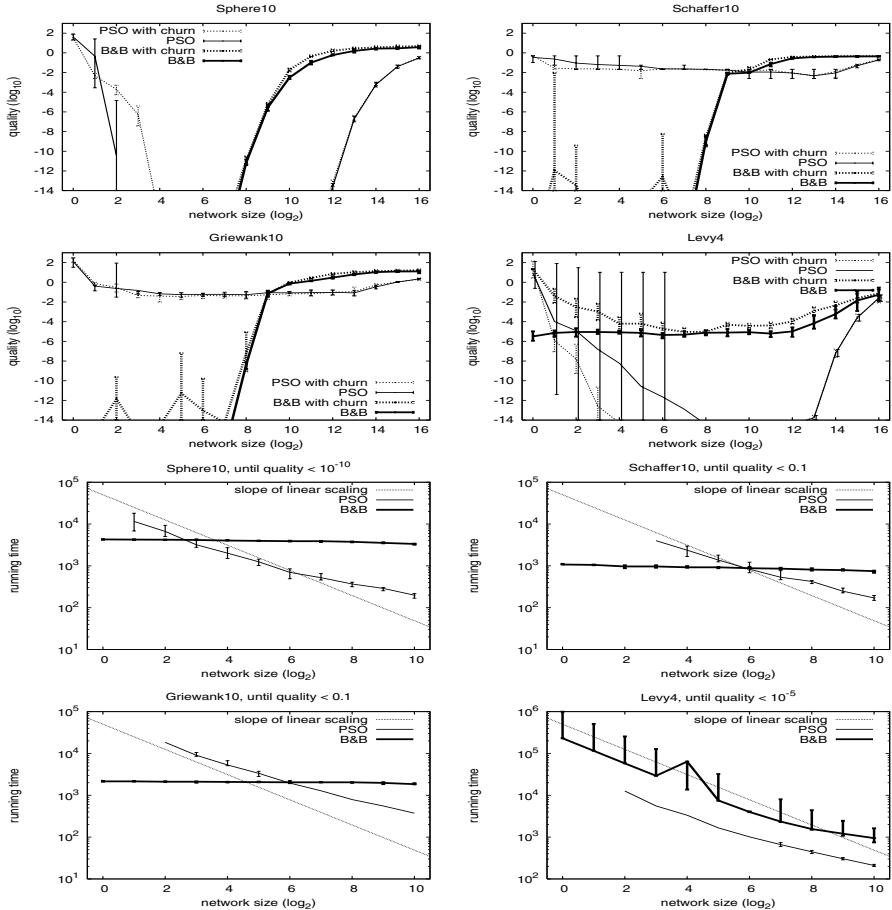


Fig. 1. For quality plots the average digits of precision, for running time plots the median is plotted, for both cases with error bars indicating the 10% and 90% percentiles of 10 experiments (100 experiments for the more unstable Levy4). Note that lower values for quality are better (indicate more precise digits). Missing error bars or line points indicate that not enough runs reached the quality threshold within 2^{20} evaluations (when the runs were terminated).

As of running time, P2P PSO always fully utilizes the network by definition, assuming a synchronous startup of the protocol, so its running time is $2^{20}/N$. In the case of P2P B&B, the situation is more complex, as illustrated by Figure 2: running time is sensitive to problem difficulty and churn.

Running time results in Figure 1 illustrate our second question on scaling: the time needed to reach a certain quality. The most remarkable fact that we observe is that on problems that are easy for B&B, it is extremely fast (and also extremely efficient, as we have seen) on smaller networks, but this effect does not scale up to larger networks. In fact, the additional nodes (as we increase network size) seem only to add unnecessary overhead. On Levy4, however, we observe scaling behavior similar to that of PSO.

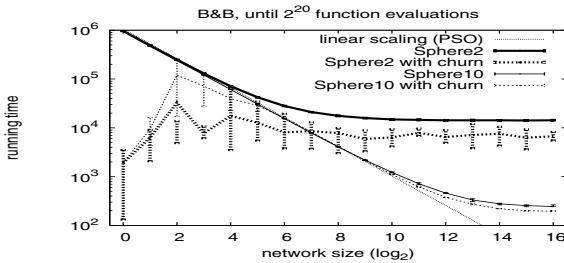


Fig. 2. Running time of P2P B&B to reach 2^{20} evaluations. Average is shown with error bars indicating the 10% and 90% percentiles of 10 experiments.

Finally, to conclude the paper, we note that unlike in small-scale controlled environments, in P2P networks system parameters (like network size and churn rate) effectively play the role of algorithm parameters, but are non-trivial to observe and exploit. An exciting research direction is to monitor these parameters (as well as the performance of the algorithm) in a fully-distributed way, and to design distributed algorithms that can adapt automatically.

References

1. Kermarrec, A.M., van Steen, M. (eds.): ACM SIGOPS Operating Systems Review 41 (October 2007); Special issue on Gossip-Based Networking
2. Biazzini, M., Montresor, A., Brunato, M.: Towards a decentralized architecture for optimization. In: Proc. of IEEE IPDPS, Miami, FL, USA (April 2008)
3. Wickramasinghe, W.R.M.U.K., van Steen, M., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: Proc. of GECCO, pp. 1460–1467. ACM Press, New York (2007)
4. Laredo, J.L.J., Eiben, E.A., van Steen, M., Castillo, P.A., Mora, A.M., Merelo, J.J.: P2P evolutionary algorithms: A suitable approach for tackling large instances in hard optimization problems. In: Luque, E., Margalef, T., Benítez, D. (eds.) Euro-Par 2008. LNCS, vol. 5168, pp. 622–631. Springer, Heidelberg (2008)
5. Jelasity, M., Montresor, A., Babaoglu, O.: A modular paradigm for building self-organizing peer-to-peer applications. In: Di Marzo Serugendo, G., Karageorgos, A., Rana, O.F., Zambonelli, F. (eds.) ESOA 2003. LNCS (LNAI), vol. 2977, pp. 265–282. Springer, Heidelberg (2004)
6. Bendjoudi, A., Melab, N., Talbi, E.G.: A parallel P2P branch-and-bound algorithm for computational grids. In: Proc. of IEEE CCGRID, Rio de Janeiro, Brazil, pp. 749–754 (2007)
7. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. ACM Transactions on Computer Systems 25(3), 8 (2007)
8. Talbi, E.G. (ed.): Parallel Combinatorial Optimization. Wiley, Chichester (2006)
9. Casado, L.G., Martinez, J.A., Garcia, I., Hendrix, E.M.T.: Branch-and-bound interval global optimization on shared memory multiprocessors. Optimization Methods and Software 23(5), 689–701 (2008)
10. Ratschek, H., Rokne, J.: Interval methods. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization. Kluwer, Dordrecht (1995)
11. PeerSim, <http://peersim.sourceforge.net/>

Evolving High-Speed, Easy-to-Understand Network Intrusion Detection Rules with Genetic Programming

Agustín Orfila, Juan M. Estevez-Tapiador, and Arturo Ribagorda

Universidad Carlos III de Madrid, Leganes 28911, Spain
`{adiaz,jestevez,arturo}@inf.uc3m.es`

Abstract. An ever-present problem in intrusion detection technology is how to construct the patterns of (good, bad or anomalous) behaviour upon which an engine have to make decisions regarding the nature of the activity observed in a system. This has traditionally been one of the central areas of research in the field, and most of the solutions proposed so far have relied in one way or another upon some form of data mining—with the exception, of course, of human-constructed patterns. In this paper, we explore the use of Genetic Programming (GP) for such a purpose. Our approach is not new in some aspects, as GP has already been partially explored in the past. Here we show that GP can offer at least two advantages over other classical mechanisms: it can produce very lightweight detection rules (something of extreme importance for high-speed networks or resource-constrained applications) and the simplicity of the patterns generated allows to easily understand the semantics of the underlying attack.

1 Introduction

This paper explores the application of GP [5] to the network intrusion detection problem, particularly to automatically creating detection patterns/rules. To the best of our knowledge, the idea is somewhat recent and was proposed for the first time in 1995 by Crosbie [2]. Since then, the most relevant works have been [9,1,4]. The problems we have identified in these previous efforts can be summarised in two main points. First, these works do not make the most of GP. The semantics of the domain is not captured in the solutions and this situation can be improved in many aspects. Most of the times the function set chosen include primitives poorly justified, such as trigonometric functions. This makes the solutions very difficult to interpret. Second, experimental work is mainly done over the well-known benchmark of KDD-99 [3]. Although this benchmark has become a standard in the field, it has certain drawbacks such us being out of date or presenting an unrealistic high prevalence of attacks. Moreover, KDD-99 dataset was created for a contest by processing the `tcpdump` portions (i.e. raw network traffic) of the 1998 DARPA IDS Evaluation Dataset, created by Lincoln Labs under contract to DARPA [6]. The original traffic generation process and the

feature extraction done for the KDD contest remain controversial [7]. Any IDS based on data-mining is very dependant upon the dataset, since the solutions are mathematical relations between the benchmark features. As a consequence, it is unclear the behaviour exhibited by these proposals in a real scenario. In addition, a serious problem that current IDSs have to face is their application in high-speed networks or, alternatively, in resource-constrained environments. A lightweight detection is required in both cases; otherwise the IDS becomes a bottleneck (and eventually it is disconnected for performance reasons), or simply it cannot be deployed in devices with shortage of computational resources. GP, however, includes a natural mechanism to limit the amount of resources consumed by the evolved individuals (e.g. limiting the number of nodes and/or depth of trees, or associating a cost to each node in a tree and penalising trees with higher costs.) This opportunity seems to have been systematically ignored in previous works. For these reasons, instead of using KDD benchmark our work focuses on publicly available raw tcp traffic from an enterprise network [8]. All in all, it is clear that all this research supports the idea that GP can be effectively applied to intrusion detection. Nevertheless we think that further work should be done with other datasets in order to avoid the dependency of the results on the dataset. Moreover, the function set should provide more understandable semantics in relation to why an event is considered an intrusion.

In this paper, we have constrained the experimental work to the capacity of GP to find analysis engines that are able to detect scanning attacks based on TCP (Transmission Control Protocol.) Therefore, traffic is only processed at the transport layer and the TCP headers are extracted to be used as inputs for our system. Results show that many TCP scans can be detected with simple rules that directly operate on the TCP header fields. Furthermore, we show that extra knowledge about the nature of the attack can be extracted thanks to the simplicity of the generated patterns and the set of operators chosen. In order to put the results in context, we compare GP solutions with those achieved by the classical machine learning algorithm C4.5 in terms of effectiveness, efficiency and semantics.

2 Design

For simplicity, in the work presented in this paper we shall restrict ourselves to analyse the capabilities of standard GP to detect a specific form of attack: scanning probes relying upon misuses of the TCP protocol. The design principles are, however, very general and can be trivially extended to other forms of network traffic and/or attacks.

For the case at hand, we extract the TCP header from each captured network packet (see Section 3 for details about the traffic source used.) The header is then processed by extracting all the fields (excluding options and checksum) and converting them into a 32-bit unsigned integer. It is important to note that

each TCP flag (URG, ACK, PSH, RST, SYN, FIN) is converted separately to an attribute. The vector resulting of this conversion is the input to the analysis engine of the IDS, which will process it according to one or more individuals obtained by GP in a training phase. Thus, the output of the engine is just 0 if the packet analysed is considered non-scanning or 1 otherwise. In what follows we describe the main GP components used in this work.

2.1 Function and Terminal Sets

The set of functions define the core of solutions in standard GP. Domain knowledge is captured through the election of these functions up to a certain extent. Therefore, it is desirable to employ functions having a clear and simple semantics, as well as a fast implementation. In our case, the chosen function set comprises of logic operators (*or_logic*, *and_logic*, *not_logic*) and bitwise operators (*or*, *and*, *not*). Moreover, the function *if_srcport_ge_10000* is also introduced to provide a different point of view in the searching process, allowing to split the instances in those with a source port over or under 10000. The set of terminals is relatively simple. As we have already stated, each TCP field, including flags separately, is represented by a 32-bit unsigned integer. We also included Ephemeral Random Constants (ERCs) for completing the terminal set. An ERC is a constant value used by GP in order to generate better individuals (for a more detailed explanation on ERCs see [5].) In our system, ERCs are 32-bit random values that can be included as terminals of the IDS.

2.2 Fitness Function

The chosen fitness function is the difference between the hit rate (H) and the false alarm rate (F) multiplied by the accuracy. The hit rate or detection rate stands for the probability of an alarm given an intrusion $P(A|I)$ while the false alarm rate stands for the probability of an alarm given no intrusion $P(A|NI)$. These quantities can be computed as follows:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total number of instances}} \quad (1)$$

$$H = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2)$$

$$F = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \quad (3)$$

2.3 Tree Size Limitations

We have constrained the maximum number of nodes to 20 nodes and the tree depth to 6. This was done in order to obtain short individuals with an easy interpretation of how they work. This restriction is adjustable, although in our experimentation these parameters have demonstrated to produce very small and sufficiently accurate rules.

3 Experimental Work

Our experiments have been done using a modern dataset¹ [8]. This dataset corresponds to internal enterprise traffic (Lawrence Berkeley National Laboratory) recorded at a medium-sized site. It was systematically anonymised [8] and only the resulting packet header traces (available in `tcpdump/pcap` format) are publicly available. Two kinds of files for the traces are provided, anonymised separately. One corresponds to the non-scanning traffic and the other to the scanning traffic. We have used the first five traces of the dataset. Both non-scanning and scanning traffic files were filtered to get rid of non TCP traces. After this process, the number of packets under consideration is 3,415,747. Some of them were used for evolving individuals and some for the testing phase, as will be explained later. Table 1 shows the number of non-scanning and scanning TCP packets in each analysed trace after the filtering process.

Table 1. Number of TCP packets in the 5 traces of the dataset considered

Trace number	Non-scanning	Scanning	Total
1	82,817	646	83,463
2	619,120	2,295	621,415
3	2,206,587	2,637	2,209,224
4	3,828	80	3,908
5	496,795	942	497,737
All	3,409,147	6,600	3,415,747

At a first stage, different initial populations were evolved using just one trace of the dataset. Afterwards, the best individual obtained was tested over the remaining traces. Trace number 3 has not been used for the evolution stage due to its large size (2,209,224 instances.)

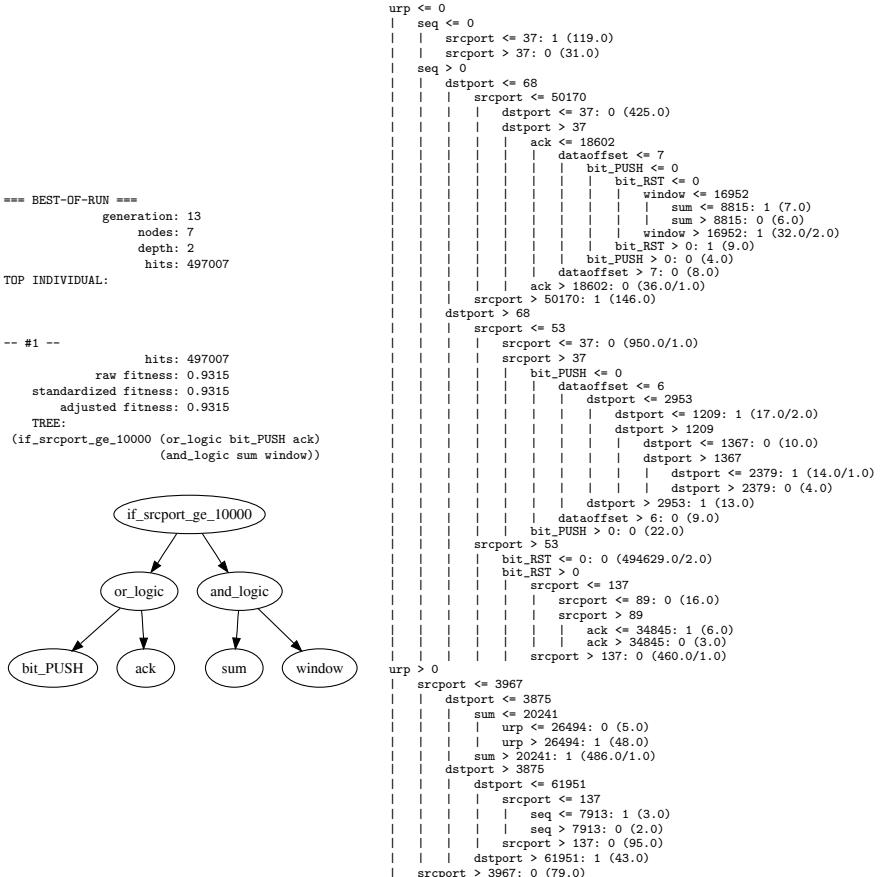
For each trace, we ran 15 experiments with 15 different seeds. The results presented below correspond to the best individuals generated after these 15 experiments, with a population size of 100 individuals, a crossover probability of 0.9, and an ending condition of 25 generations. Results are compared with those obtained by the machine learning algorithm C4.5. In order to make a fair comparison, the same process was followed in the case of C4.5: one of the five traces was used to build the model and the remaining for testing purposes.

Table 2 summarises the best results obtained by GP and C4.5. It is important to note that the prevalence of scanning TCP packets in the first five traces of the dataset is 0.0019. Thus, a naive IDS stating that every TCP packet is non-scanning would achieve a 99.81% of accuracy (with $H=0$ and $F=1$.) Therefore, when the distribution of the classes (scanning, non-scanning) is so unbalanced, accuracy is not a good measure of effectiveness. Accordingly, we show the results providing H and F . Thus, Table 2 shows that GP provides more effective rules than C4.5 for models derived from traces 1 and 5, but not for those from traces 2 and 4. The C4.5 model and the best GP individual obtained from trace 5 are represented in Figure 1.

¹ LBNL Enterprise trace Repository, <http://www.icir.org/enterprise-tracing/>

Table 2. Summary of results

Trace used for evolution	Testing traces	GP		C4.5	
		fitness = $(H - F)$	Accuracy	F	H
1	2,3,4,5	0.1876	0.8013	0.0809	0.2057
2	1,3,4,5	0.0155	0.8546	0.0014	0.7936
4	1,2,3,5	0.0016	0.2979	0.0028	0.5304
5	1,2,3,4	0.0041	0.5322	0.0021	0.4231

**Fig. 1.** Comparison of GP top individual (left) and C4.5 model (right) obtained from trace 5

As can be seen, GP solution is much simpler than C4.5 one (7 vs. 63 nodes.) Thus, it is easy to understand why the generated GP rule detects a scan. In contrast, the C4.5 tree is complex enough (63 nodes) to make it difficult to comprehend the nature of the detection. This is clearly an advantage in favour of GP technique.

Table 3. Efficiency of GP vs. C4.5: Number of basic operations per TCP packet

Trace used for evolution	GP <i>fitness</i> = $(H - F)$ Accuracy	C4.5
1	12	28
2	18	33
4	11	4
5	7	40

Apart from semantics reasons, the complexity of the rules has an impact on the detection efficiency. Table 3 shows a comparison in terms of the number of basic operations per TCP packet required for the worst case. The *and*, *or*, *not*, *and_logic*, *or_logic*, *not_logic* functions and load operations are considered basic (i.e. with a cost of 1 unit), while *comparisons* are assumed to be formed by two basic operations. GP models are generally much more efficient than C4.5 ones (the only exception corresponds to trace 4 due to its small size—see Table 1.)

References

1. Abraham, A., Grosan, C., Martin-Vide, C.: Evolutionary design of intrusion detection programs. International Journal of Network Security 4(3), 328–339 (2007)
2. Crosbie, M., Spafford, E.H.: Applying genetic programming to intrusion detection. In: Working Notes for the AAAI Symposium on GP (1995)
3. Elkan, C.: Results of the KDD 1999 classifier learning contest (September 1999), <http://www-cse.ucsd.edu/users/elkan/clresults.html>
4. Faraoun, K., Boukelif, A.: Genetic programming approach for multi-category pattern classification applied to network intrusions detection. The International Arab Journal of Information Technology 4(3), 237–246 (2007)
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
6. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks 34(4), 579–595 (2000)
7. Mahoney, M.V., Chan, P.K.: An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 220–237. Springer, Heidelberg (2003)
8. Pang, R., Allman, M., Bennett, M., Lee, J., Paxson, V., Tierney, B.: A first look at modern enterprise traffic. In: Proceedings of the 5th ACM SIGCOMM conference on Internet measurement, IMC 2005, pp. 1–14. ACM, New York (2005)
9. Song, D., Heywood, M.I., Zincir-Heywood, A.N.: Training genetic programming on half a million patterns: an example from anomaly detection. IEEE Transactions on Evolutionary Computation 9(3), 225–239 (2005)

Soft Computing Techniques for Internet Backbone Traffic Anomaly Detection

Antonia Azzini¹, Matteo De Felice^{2,3}, Sandro Meloni³,
and Andrea G.B. Tettamanzi¹

¹ University of Milan, Information Technology Department
{azzini,tettamanzi}@dti.unimi.it

² ENEA (Italian Energy New Technology and Environment Agency)

³ University of Rome “Roma Tre”, Department of Informatics and Automation
{defelice,sandro}@dia.uniroma3.it

Abstract. The detection of anomalies and faults is a fundamental task for different fields, especially in real cases like LAN networks and the Internet. We present an experimental study of anomaly detection on a simulated Internet backbone network based on neural networks, particle swarms, and artificial immune systems.

1 Introduction and Problem Definition

In computer networks, anomaly detection is currently one of the hottest topics, whose unambiguous goal is searching for potential security breaches. The study of techniques for the detection and prevention of traffic anomalies has spanned a rich spectrum of paradigms. Nonetheless, it still represents a difficult challenge, not only because the Internet infrastructure is not designed to detect anomalies, but also because volume anomalies can take a wide range of different forms, each in principle characterized, at the onset, by a different traffic profile.

The problem of anomaly detection can be naturally recast into a classification problem. In such a scenario, several approaches have been presented in the literature, with an increasing interest in machine learning and in the application of nature-inspired algorithms, such as evolutionary algorithms (EA), artificial neural networks (ANN), and artificial immune systems (AIS). This work discusses some possible applications of such techniques to anomaly detection, showing the most important aspects of the considered methodologies applied to two examples of fault cases.

One of the most typical examples of anomaly, related to computer security, regards Network Intrusion, an attack to a system coming through the network where computers communicate via standard protocols. Other typical examples of anomalies are traffic volume anomalies, generated, for example, by denial-of-service (DoS) attacks or flash crowds. They consist in a large surge in traffic to a particular site causing a dramatic increase in server load and putting severe strain on the network links leading to the server, which, in turn, results in a considerable increase of packet loss and congestion, degrading network operation and impacting the perceived quality of service (QoS) for the user [11]. It is therefore important to be able to timely detect them from the onset.

The remainder of this paper is organized as follows. Section 2 describes the detector models and the algorithms implemented in this work, while section 3 presents the experiments carried out and discusses the results obtained. Finally, Section 4 provides some concluding remarks.

2 Detector Models and Detector Set Generation

In the anomaly detection problem, different kinds of models are used to define detectors. Some make use of ANNs [10], while others consider geometric coverages by using, for example, hyperspheres, hypercubes, or hyperellipses [1,3]. In this work, we consider three types of detectors: two kinds of ANNs, respectively feed forward (FFNs) and radial basis functions networks (RBFNs), and hypersphere sets:

- *Hyperspherical Detectors* consisting of a set of coordinates, representing a point in the n -dimensional feature space, and a radius value (hyperradius).
- *Artificial Neural Networks*, able to distinguish between normal and anomalous pattern after observing a set of pattern during a learning phase (*training phase*). The classification obtained is an index that varies from **normal** to **anomalous**, which we mapped respectively to the real values 0 and 1. In this work Feed-Forward Networks (FFNs) and Radial Basis Function Networks (RBFNs) are considered, described in detail in [7].

Three different approaches are implemented for optimal detector definition: the usual error backpropagation algorithm for training FFNs, Particle Swarm Optimization (PSO) for optimizing hyperspherical detectors, and negative selection (NS) of Artificial Immune Systems. The main features are reported below, while the specific parameter settings used in this work are reported in Table 1.

Particle Swarm Optimization is a population-based optimization technique [2,12], whereby each individual moves in the solution space following two main attraction points: the best solution it has encountered so-far and the best solution found by any other individual in the swarm at a given time.

Only normal cases are considered and the fitness function is defined as the ratio between the true positive range of normal cases for each detector (also indicating the density of each detector) and its area, penalizing possible overlaps between detectors in the positive recognition.

A modified version of the traditional PSO is implemented in this work in order to speed up convergence. Low-fitness detectors (i.e., low density of normal traffic detected) are attracted to the best ones, while the worst detectors, with no normal coverage, are reinitialized. Furthermore, detector diversity is maintained through a *repulsion* mechanism, in order to reduce their overlaps. Finally, at each iteration, while the fitness does not worsen, the radius of each detector is reduced (*radius self-tuning*). Such modified PSO allows to avoid hypersphere detectors with too large radii, thus improving the coverage of the normal space and reducing holes.

Neural Networks Training: NNs are used for the non-anomalous space covering. Generally, for real problems, the set of positive (non-anomalous) instances

is smaller than the complementary set of anomalous instances. For this reason, in this work we train ANNs both with the positive set and some randomly-generated noise patterns, as fault cases.

Negative Selection: is one of the major algorithms developed in the field of AIS [8]. It generates a set of detectors covering the complementary space to the normal one, in order to classify new, unseen data as normal or anomalous. The detector set is defined through randomly generated detectors that do not recognize normal samples as anomalous.

3 Experiments

In order to compare the behavior of the considered detector models, all the experiments carried out in this work refer to anomalous events in a heavy-traffic network, the Abilene Internet 2 Backbone¹, by observing signals on a subset of the network's nodes and links. A network simulator is used to generate the dataset of the network traffic signals.

The Abilene Internet 2 network is a US continental backbone, connecting several educational, research, and commercial institutions. The main backbone consists of 9 macro-nodes connecting some of the main US cities and 14 OC-192 (10-gigabit ethernet optical fiber cable) links. The average traffic volume is about 1 Gb/s over the links, with two different kinds of periodical oscillations, representing respectively daily (day/night) and weekly fluctuations (week end/working days). In the Abilene network modeling, efforts have been spent to reproduce the typical traffic volumes and shapes over the links starting from an Origin/Destination Matrix (OD-Matrix). Although this is known as a generally NP-hard problem, an approximate solution can be obtained by studying the topology and the behavior of the network.

The software used to define the network is Network Simulator 2 (NS2)², that reproduces all the traffic features as the TCP/IP protocol stack, and it is able to generate traffic information qualitatively and quantitatively comparable with the actual Abilene backbone. Two different link fault cases have been created with this simulator for the experiments.

Table 1 shows the most important parameters considered for each model, together with the values that they take up in the experiments carried out. All the detector models are tested over 10 runs for each parameter setting. The bold value for each model corresponds to the best setting.

3.1 Results and Discussion

Two groups of experiments are carried out in this work, by considering, respectively, the flow through the nodes and through the links of the simulated network. The results of the latter group of experiments are shown in detail, since they

¹ <http://www.internet2.edu/network/>

² <http://www.isi.edu/nsnam/ns/>

Table 1. Parameter Settings for the detector models: FFNs with the backpropagation algorithm (**BP-FFNs**), RBFNs with the training algorithm (**RBFNs**), Negative Selection with FFNs (**NS-FFNs**), Negative Selection with RBFNs (**NS-RBFNs**), Negative Selection with Hyperspherical Detectors (**NS-HDs**), PSO with Hyperspherical Detectors (**PSO-HDs**). The bold values for each model correspond to the best settings.

ANNs	BP-FFNs	Learning Rate: Adaptive ; Training: Resilient Backpropagation ; Transfer functions: [logsig - purelin], [tansig - purelin]; Network topology: [3-3-1] ; Stopping criterion: max epochs (1000), minimum gradient ($1e^{-10}$)
	RBFNs	Training: MATLAB newrb ; Maximum number of neurons: 20,30,50
PSO	PSO-HDs	Maximum number of iterations: 200,500 ; Number of particles: 10, 25 ; Number of detectors modeled by each particle: 5, 8 ; Radius Self-Tuning: yes,no ; Repulsion: yes,no
NS	NS-FFNs	Number of detectors: 10 ; Transfer functions: [logsig - purelin], [tansig - purelin]; Network topology: [3-3-1] ; Stopping criterion: max epochs (1000) / minimum gradient ($1e^{-10}$); Acceptance threshold: 0.2
	NS-RBFNs	Number of detectors: 10 ; Spread range: [100,700], [300,700] ; Detectors radius range: [10,250], [10,300] ; Acceptance threshold: 0.2
	NS-HDs	Number of detectors: 100, 150 ; Detectors Radius: variable, costant

present more interesting and more effective results over the considered traffic network. In the link-flows, the variations are less dramatic after the link fault, especially if we use links not strictly involved with the fault as inputs. Figure 1 shows the results obtained from the simulated fault case of the link between the cities of Chicago and Atlanta.

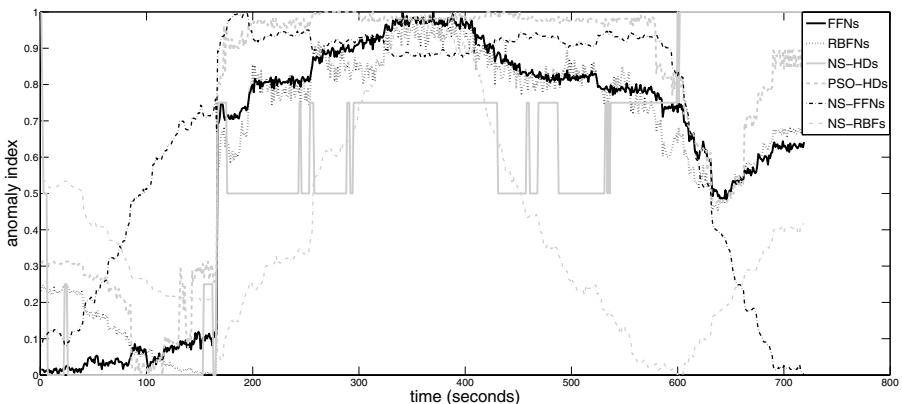


Fig. 1. Comparison of the best results obtained from the detector models

In this case, after the link fault, the traffic routing changes and does not revert to the previous situation even after the links come back up. Indeed, we can see in Figure 1 that after the fault around second 160, the anomaly index remains high for a very long period, even if the fault ends at time 250. The only technique which does not detect the fault properly is **NS-RBFNs**, while the others show a visible edge around the fault event, although the edge of **NS-FFNs** is smooth and small.

ANNs show good performances on all the experiments conducted, especially FFNs, representing a really common and well-studied tool for a wide variety of applications. Nevertheless, the fact should be emphasized that ANN training requires a complete dataset with instances of both classification classes, i.e., normal and anomalous data, which usually is not available, because a complete characterization of all the possible anomalies is not available in general.

We consider the performance of PSO (PSO-HDs) really encouraging: our modified version of the PSO algorithm is able to detect almost all of the faults presented, even when considering an offline anomaly detection problem only. Indeed, the nature of this algorithm makes it a convenient choice for real-time detection systems and well-suited for non-stationary environment. The main advantage of this technique is the easiness of implementation and extension with new functions, as well as its fast execution time with any coding language.

NS-based methods are usually a good choice for anomaly detection problems. We observe in our experiments good performances both with hyperspherical detectors and with ANNs, even if the latter need a more accurate parameters tuning, a characteristic often observed with ANNs.

The models considered in this work have been widely applied in different approaches presented in the literature regarding the anomaly detection problem. One of the first observations carried out in the survey by Kim and colleagues [13] is that various features of the NS algorithm make it by far the most popular algorithm for solving such problems. However, despite its appealing properties, it has not shown great success in real-life applications. The authors indicate two drawbacks to utilizing the NS algorithm, namely scalability and coverage, defining them the main barriers to their success as an effective detection model.

Timmis and Freitas [9] too indicate the use of a NS-based AIS as problematic, presenting, besides those indicated in [13], other disadvantages; indeed, the random generation of detectors is not adaptive and does not use any information to guide search. Then, the lack of mechanisms to minimize overfitting, and the fact that NS is mainly an algorithmic rather than a problem-oriented approach, produce poor solutions in the model definition.

In the literature, analogies with other intelligent techniques were and are still today appropriate ideas in order to improve NS in anomaly detection problems, with particular attention to evolutionary approaches, like genetic algorithms and PSO, artificial neural networks and, in some cases, also fuzzy rules. Furthermore, joint negative and positive selection could be a satisfactory solution. Hybrid representations that use evolutionary approaches in analogy with AIS could become useful in order to achieve a high level of robustness and adaptability. Moreover, solutions implemented with neural networks require a reduced number of detectors *vis à vis* those that use geometrical detectors, also by considering simple topologies, thus reducing the overall computational cost.

4 Conclusion and Future Work

In this work, a particular attention has been paid to applications based on nature-inspired algorithms for a normal vs. anomalous network traffic classification.

Different detector models have been applied, together with different detection algorithms, to two simple cases of fault detection. The results obtained from the experiments carried out show how all such techniques may obtain satisfactory results even without an in-depth preliminary analysis and tuning of the parameters for each implemented algorithm.

Future works will consider more difficult network traffic situations, with more irregular traffic data (by considering, for example, LAN traffic information), or real-time fault detection. Moreover, other algorithms should be considered for network traffic analysis, for example CLONALG [5] and Immune Networks [6].

References

1. Balachandran, S., Dasgupta, D., Nino, F., Garrett, D.: A framework for evolving multi-shaped detectors in negative selection. In: Proc. of IEEE Symposium on Foundations of Computational Intelligence, FOCI 2007, pp. 401–408 (2007)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm intelligence: From natural to artificial systems*. Oxford University Press, Oxford (1999)
3. Bouvry, P., Seredytsky, F.: Anomaly detection in TCP/IP networks using immune systems paradigm. *Computer Comm.* 30, 740–749 (2007)
4. Dasgupta, D., Ji, Z.: Real-valued negative selection algorithm with variable-sized detectors. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 287–298. Springer, Heidelberg (2004)
5. De Castro, N.L., von Zuben, F.J.: Learning and optimization using the clonal selection principle. *IEEE Trans. on Evolutionary Computation* 6(3), 239–251 (2002)
6. De Castro, N.L., von Zuben, F.J.: Immune and neural network models: Theoretical and empirical comparisons. *Int. Journal on Computational Intelligent Applications* 1(3), 239–257 (2001)
7. Dreyfus, G.: *Neural networks, methodology and applications*. Springer, Heidelberg (2005)
8. Forrest, S., Perelson, A., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proc. of the IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, pp. 202–212 (1994)
9. Freitas, A.A., Timmis, J.: Revisiting the Foundations of Artificial Immune Systems for Data Mining. *Trans. on Evolutionary Computation* 11(4) (August 2007)
10. Gao, X.Z., Ovaska, S.J., Wang, X., Chow, M.Y.: A neural networks-based negative selection algorithm in fault diagnosis. *Neural Computing & Applications* 17, 91–98 (2007)
11. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In: Proc. of WWW 2002, Hawaii (May 2002)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of IEEE International Conf. on Neural Networks, Perth, Australia, vol. 4, pp. 1942–1948 (1995)
13. Kim, J., Bentley, P.J., Aickelin, U., Greensmith, J., Tedesco, G., Twycross, J.: Immune system approaches to intrusion detection - a review. *Natural Computing* 6, 413–466 (2007)

Testing Detector Parameterization Using Evolutionary Exploit Generation

Hilmi G. Kayacik¹, A. Nur Zincir-Heywood¹, Malcolm I. Heywood¹,
and Stefan Burschka²

¹ Faculty of Computer Science, Dalhousie University,
6050 University Avenue, Halifax NS, B3H 1W5, Canada

² Software & Security Technologies, Swisscom Innovations, Switzerland
`{kayacik,zincir,mheywood}@cs.dal.ca,`
`Stefan.Burschka@swisscom.com`

Abstract. The testing of anomaly detectors is considered from the perspective of a Multi-objective Evolutionary Exploit Generator (EEG). Such a framework provides users of anomaly detection systems two capabilities. Firstly, no knowledge of protected data structures need be assumed. Secondly, the evolved exploits are then able to demonstrate weaknesses in the ensuing detector parameterization. In this work we focus on the parameterization of the second generation anomaly detector ‘pH’ and demonstrate how use of an EEG may identify weak parameterization of the detector.

1 Introduction

Buffer overflow attacks represent one of the most effective and widespread examples of a host based attack. Detector techniques for recognizing such attacks generally fall into two basic forms: misuse or anomaly. The misuse paradigm focuses on recognizing the attack, whereas the anomaly paradigm concentrates on modeling ‘normal’ behaviour and flags everything else as ‘anomalous.’ Given the ease with which current examples of misuse detection can be evaded [1], we concentrate on the case of anomaly detection. Previous work on evolving attacks has indicated that it is possible to evolve exploits against the original Stide anomaly detector [2] when feedback from the detector is limited to the (publicly available) alarm rate [3]. Moreover, such works only consider crafting an exploit, without the preamble which set up the vulnerability such that the exploit can be launched.

Unlike the original Stide anomaly detector, the second generation process Homeostasis (pH) model of anomaly detection both detects and instigates counter measures against suspicious processes [4]. Detection is performed relative to sequences of system calls made by an application and processes with above normal anomaly rates receive a corresponding exponentially weighted delay. The goal of this work is to perform a vulnerability assessment of the pH anomaly detector under a vulnerable UNIX application using a black box approach. Specifically, the only detector information used to ‘guide’ the search for better exploits is the alarm

rate, where this is public information available to a user. Conversely, in order to establish better criteria for EEG we make use of a Pareto multi-objective fitness function. We demonstrate that such a framework presents users with a wider characterization of the scope of potential vulnerabilities for a given anomaly detector, as well as quantifying the impact of the detector parameterization. Moreover, we explicitly include the impact of the preamble within the evaluation, so that exploits are identified within context.

2 Methodology

We are interested in evolving buffer overflow attacks against pH where the detector has the capacity to delay the process associated with suspicious system call sequences. Moreover, given that an attack is composed from preamble and exploit it would also be useful to reward the evolutionary exploit generator for reducing anomaly rates associated with the attack as a whole as opposed to the exploit alone. Given the multiple criteria associated with this task it is reasonable to assume a Pareto framework for designing our exploit generator. Before we do so, however, we first need to establish the basic components of the evolutionary model; in particular the representation and model of deployment. To this end we assume the framework established by Kayacik *et al.* [3] and summarize this in Section 2.1. The design of the multi-criteria fitness function is considered relative to the basic framework in Section 2.2.

2.1 Basic Evolutionary Exploit Generator

A basic framework for EEG under a black box detector assumption was considered to comprise of two basic steps [3]: identification of a representative set of system calls, and design of the fitness function, as follows:

System Calls: It is necessary to *a priori* specify a set of instructions (system calls) from which candidate exploits are built. The approach adopted by Kayacik *et al.* was to first run the vulnerable application (*not* the anomaly detector) under normal operation and then review the most frequently executed system calls during execution. That is to say, a diagnostic tool – in this case Strace¹ – is deployed to establish the most frequent 20 system calls of the application (accounts for upwards of 90 percent of the system call utilization). It is these system calls that are used as the instruction set of the evolutionary exploit generator.

Fitness Function: Two basic criteria were previously identified for guiding evolution towards an effective attack: criteria for a valid attack, and quality of the attack. Criteria for a valid attack assumed that the basic goal of a buffer overflow exploit were to: open a file, write to the file, and close the file. The file in question was the password file, where the arguments to the open-write-close sequence established the specifics of the target file. Naturally,

¹ Strace can be downloaded from <http://sourceforge.net/projects/strace/>

the sequence of these three operations is significant. Thus, individuals were rewarded for both the instructions executed and the order of execution. In the case of exploit quality, use was made of the anomaly rate returned by the detector. Thus if the detector considered the instruction sequence anomalous the ensuing penalty would be greater.

In the following we assume the same process for identifying the subset of system calls to compose attacks. However, the criteria for designing the fitness function will now assume an Evolutionary Multi-criteria Optimization (EMO) model in order to address the increased functionality apparent in pH, but without referring to knowledge of its operational features.

2.2 Evolutionary Multi-criteria Exploit Generator

In assuming an EMO basis for evolving exploits we naturally make use of the method of Pareto ranking to combine multiple objectives into a single ‘scalar’ fitness function [5]. Specifically, a process of pairwise comparison is used to establish to what degree each individual is dominated by other individuals in the population. Thus, one individual (A) is said to dominate another (B) iff A is as good as B on all objectives and better than B on at least one. However, a wide range of algorithms have been proposed to support such a rank based fitness function [6]. In this work we assume the PCGA framework of Kumar and Rockett [7]. Such a scheme avoids the need to support explicit measures of similarity, where such schemes are generally employed to encourage diversity and/or measure similarity in the population. Instead diversity is established care of a steady-state style of tournament. Specifically, after the initial population is ranked, parents are chosen using proportional selection and the ensuing children ranked. The rank of the children is compared to that of the worst ranked individuals in the population. If the children have a better rank, then they over write the (worst ranked) population members. Use of such a EMO model is necessary in this work as it is not possible to establish distance information relative to the original problem domain. That is to say, popular algorithms such as NSGA-II and SPEA all make use of Euclidean based distance functions when building the diversity metric (see for example [6]). In this work the representation takes the form of system call sequences as opposed to points in multi-dimensional space, thus precluding the application of such distance metrics.

Specific objectives measured to establish fitness of each individual under the EMO model take the form of the following three generic objectives:

1. Attack Success: As established in Section 2.1 the basic functionality of the attack is described in terms of an ‘open-write-close’ sequence of system calls, with reward established for each of the three instructions and the relevant order². A behavioural success function gives a maximum of 5 points the correct ‘open-write-close’ behaviour, Figure 1.

² Relevant arguments are also considered, although pH does not consider them.

2. Anomaly Rate: Again as established in Section 2.1, the detector anomaly rate represents the principle metric for qualifying the likely intent of a system call sequence; a would be attacker naturally wishes to minimize the anomaly rate of the detector. Again, no inside knowledge is necessary as detectors provide alarm rates as part of their normal mode of operation. Moreover, as indicated in the introduction, the attacker also needs to minimize the anomaly rate of the exploit. We evolve exploit with preamble appended to facilitate identification of the most appropriate content.
 3. Attack Length: Attack length appears as an objective to encourage evolution to perform a wider search for solutions.
1. Count = 0;
 2. IF (sequence contains *open*(“/etc/passwd)) THEN (Count++)
 3. IF (sequence contains *write*(“toor::0:root:/root:/bin/bash”)) THEN (Count++)
 4. IF (sequence contains *close*(“/etc/passwd”)) THEN (Count++)
 5. IF (‘*open*’ precedes ‘*write*’) THEN (Count++)
 6. IF (‘*write*’ precedes ‘*close*’) THEN (Count++)

Fig. 1. Fitness objective quantifying the exploit functionality

3 Results

3.1 pH Detector Configuration

pH is a second generation anomaly detector in which specific instances of the detector are associated with each application. During training, pH employs a sliding window to establish the normal behaviour of the application in terms of a matrix with dimensions: (1) current system call; (2) previous system call; (3) location of the previous system call in the sliding window. During testing, the same sliding window is employed on the candidate exploits. If a given sliding window sequence produced a look ahead pair that is not in the normal database, a mismatch is recorded. Given a pre-specified window size and system call trace length, the anomaly rate for the trace is calculated by dividing the number of mismatches by the total number of look ahead pairs. Moreover, the delay property used by pH to penalize suspicious processes is calculated as an exponential function of locality frame count; where locality frame (LF) count aims to identify clusters of anomalies. In this work, the default pH training parameters were employed, as defined by Somayaji [4].

3.2 Traceroute Application Configuration

In the following experiments, traceroute is tested. Traceroute is a vulnerable application frequently used in the attack automation literature [8]. Traceroute is a network diagnosis tool, which is used to determine the routing path between a source and destination by sending a set of control packets to the destination with increasing time-to-live values. Redhat 6.2 is shipped with Traceroute version 1.4a5, where this is susceptible to a local buffer overflow exploit that

provides a local user with super-user access [9]. In order to establish traceroute behavior under normal conditions we developed five use cases: (1) targeting a remote server, (2) targeting a local server, (3) targeting a non-existent server, (4) targeting the localhost, (5) help screen.

3.3 Performance Evaluation

Each run of the EEG may provide up to 500 exploits. In the following analysis, we concentrate on the non-dominated performance characteristics as captured post training. Specifically, this will take the form of three properties: anomaly, sequence length, and delay. By way of a baseline we also include the performance of the original attack. Table 1 details the characteristics of the non-dominated attacks under the Traceroute application. The last row details the characteristics of the original attack. It is apparent that the anomaly rate of the overall attack can be significantly reduced. In terms of attack and exploit delays, the attack that achieves the least delay achieves this by deploying a short exploit, hence reducing the delay from 6.39E+06 to 0.55 seconds. Naturally, this is only achieved while doubling the anomaly rate of the attack (from 16.29% to 30.91%).

Table 1. Non-dominated characteristics of Attacks against Traceroute application

Optimization Criteria	Attack Anomaly (%)	Exploit Anomaly (%)	Length	Attack Delay (sec)	Exploit Delay (sec)
Anomaly rate	18.29	11.71	118	6.39 e+6	1.11
Delay	30.91	100	9	0.55	0.02
Length	65.11	66.77	1,000	3.83 e+28	3.75 e+28
Original	66.27	73.91	261	4.39 e+35	4.39 e+35

4 Conclusion

A second generation anomaly detection system – pH – is assessed for effectiveness under a black box information criterion. To do so, a multi-objective model of EEG is employed. Such an approach enables us to determine the cross-section of (non-dominated) behaviours evolved post training. It is clear that the preamble plays a very significant role in establishing the effectiveness of the overall attack. The parameterization of pH is a tradeoff between collecting statistics over longer window sizes and establishing a good model for normal behaviour versus detection of very short attacks. Under Traceroute the low system call count of the preamble enables the configuration of attacks that on the one hand register a high anomaly rate, yet escape the delay property of pH. Specifically, the total system call count for the evolved attack is 91, thus smaller than the locality frame window size of 128. Moreover, the exploit alone required 9 system calls (100 percent anomaly); thus, the use of a Pareto multi-objective model of evolution enabled us to discover that such an attack could still be very effective. Thus, the would-be attacker would have compromised the system before an administrator could have reacted to the alarm. From the perspective of the overall vulnerability analysis the EEG enables us to optimize the detector parameterization.

Acknowledgements

The authors gratefully acknowledge the support of SwissCom Innovations, MITACS, CFI and NSERC grant programs.

References

1. Kayacik, H.G., Heywood, M., Zincir-Heywood, N.: On evolving buffer overflow attacks using genetic programming. In: Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO), SIGEVO, pp. 1667–1674. ACM, New York (2006)
2. Forrest, S., Hofmeyr, S.A., Somayaji, A.B., Longstaff, T.A.: A sense of self for unix processes. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 120–128 (1996)
3. Kayacik, H.G., Heywood, M., Zincir-Heywood, N.: Evolving buffer overflow attacks with detector feedback. In: Giacobini, M. (ed.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 11–20. Springer, Heidelberg (2007)
4. Somayaji, A.B.: Operating system stability and security through process homeostasis. PhD thesis, The University of New Mexico (2002)
5. Goldberg, D.E.: *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Reading (1989)
6. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, Chichester (2001)
7. Kumar, R., Rockett, P.: Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution. *Evolutionary Computation* 10(3), 283–314 (2002)
8. Tan, K., Killourhy, K., Maxion, R.: Undermining an anomaly-based Intrusion Detection System using common exploits. In: Wespi, A., Vigna, G., Deri, L. (eds.) *RAID 2002*. LNCS, vol. 2516, pp. 54–73. Springer, Heidelberg (2002)
9. SecurityFocus: Lbnl traceroute heap corruption vulnerability (last accessed June 2008), <http://www.securityfocus.com/bid/1739>

Ant Routing with Distributed Geographical Localization of Knowledge in Ad-Hoc Networks

Michał Kudelski^{1,2} and Andrzej Pacut^{1,2}

¹ Institute of Control and Computation Engineering, Warsaw Univ. of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland

m.kudelski@elka.pw.edu.pl, A.Pacut@ia.pw.edu.pl

² NASK

Wawozowa 18, 02-796 Warsaw, Poland

Abstract. We introduce an alternative way of knowledge management for ant routing in ad-hoc networks. In our approach, the knowledge about paths gathered by ants is connected with geographical locations and exchanged between nodes as they move across the network. The proposed scheme refers to the usage of a pheromone by real ants: the pheromone is left on the ground and used by ants in its surroundings. Our experiments show that the proposed solution may improve the overall performance of the underlying ant routing mechanism.

1 Introduction

The aim of this paper is to introduce an innovative way of managing the knowledge gathered by the ant routing mechanism. In our approach, we utilize the information about the nodes location to connect this knowledge with geographical locations. Knowledge is distributed yet logically stored in geographical locations and exchanged between the nodes as they move across the network. We investigate the influence of the geographical localization of knowledge on the underlying ant routing mechanism in a set of simulation experiments performed for different communication scenarios. We show that the proposed approach may improve the networks performance in terms of end-to-end transmission delay and delivery ratio.

The problem of routing is a basic issue in a communication network. In recent years, a large number of ad-hoc routing algorithms have been proposed (see [1,3] for overviews). There are many approaches based on learning mechanisms that solve problems in ad-hoc networks [8]. There are also approaches to routing known that utilize a geographical information, such as a GPSR geographical routing protocol [4] or a GRID protocol [9] with some form of geographical localization of the routing information. None of the above mentioned works, however, considers the learning mechanism with geographical localization of knowledge or addresses the problem of distribution of information. In particular, the problem of exchanging the knowledge between the nodes is not analyzed and the influence on the learning process is not investigated.

2 Geographical Localization of Knowledge

The communication medium used in wireless networks implies the relation between the space and the local network state. Nodes located in the same neighborhood share the communication medium and consequently, observe a similar network state. In our approach, described in [7], we want to take advantage of this property by connecting the knowledge with geographic locations in the network. By the knowledge we mean the information obtained and used in the learning process. We represent geographical locations with *geographical cells*. Instead of defining the information about paths on a node level, we introduce the notion of the *virtual path* defined on a location level. During the routing process, packets are directed through intermediate locations using geographical policy. We also change the way that the routing information is stored. In order to keep the connection between the knowledge and locations, the knowledge must be logically stored in the locations. As nodes in the network change their positions, they also switch to the information corresponding to the particular location. To achieve this, we propose an information exchange mechanism that allows to store the information physically in the nodes and to exchange it between the nodes, keeping logical connection between the knowledge and locations (Sec. 3).

AntHocGeo Algorithm. AntHocGeo, proposed and discussed in detail in [7], is a modification of AntHocNet, which implements the concept of geographical localization of knowledge. Simulations performed in [6,7] showed that the geographical localization of knowledge may improve the performance of ant routing. However, the experiments were performed on a simplified implementation, storing the information in a single “abstract node” placed in each geographical cell.

3 Exchanging Knowledge between the Nodes

In this section we propose a mechanism that allows nodes of an ad-hoc network to exchange their knowledge in order to realize the concept of geographical localization of knowledge. We base our proposal on the following assumptions. Knowledge is physically stored in nodes yet it is logically stored in geographical locations. Nodes within the same location synchronize the corresponding knowledge. When a node moves to another location, it obtains the corresponding knowledge from the nodes existing in the new location. Each node may store more than one instance of knowledge: not only it stores the knowledge corresponding to its current location, but also provides a backup to neighboring locations in case they are abandoned by nodes.

There are two situations when the information is proposed to be exchanged between the nodes. The first one is implied by the learning algorithm. When one of the nodes updates its knowledge as a result of the operation of the algorithm (e.g. on the basis of the information gathered by an ant), other nodes within the same location should also update their instance of the corresponding knowledge. This can be easily achieved by broadcasting or by „eavesdropping” in the MAC

layer. However, there is always a risk that some nodes may not receive some of the update messages. The second situation of information exchange is caused by the nodes changing their location. When a new node appears in a location, it sends a request for the information to other nodes in the location. It obtains several versions of the local knowledge and must create a pooled version. The trouble is that the obtained versions may differ.

Considering an ant routing mechanism, we propose the following model of updates monitoring, visualized in Table 1 as a *contact matrix*. Suppose that there are N nodes in a given location and after the last synchronization there were K updates of the given information entry, each one caused by an ant. Node i receives update j with the probability $p_{i,j}$, hence $k_{i,j} = 1$ with probability $p_{i,j}$ (update received) or $k_{i,j} = 0$ with probability $(1 - p_{i,j})$ (update missed). The local information in node i is represented by m_i , which is the estimated quality of a given path, computed as a (running) average of a value of a metric collected by ants (each ant carries a single update information in a form of the measured metric value m_j). To create a full image of the information, it would be necessary to exchange the full history of contacts between the nodes since the last synchronization. This would dramatically increase the memory requirements of the algorithm and the overhead generated by the knowledge exchange mechanism. Thus, it would be desired to utilize only the basic, averaged information m_i available in each node.

Basing only on the values m_i , it is possible to find the synchronized information m as the weighted average (see Table 1, right): $m = \frac{\sum_{i=1}^N w_i m_i}{\sum_{i=1}^N w_i}$. Weights w_i should consider the dependencies between each information instance m_i : some update values m_j may be included in more than one averaged value m_i . If the number k_i of the contacts received in each node is known and we assume that the probability $p_{i,j}$ is constant in time, we obtain: $k_i = \sum_{j=1}^K k_{i,j}$ and $p_{i,j} = p_i = k_i/K$. We propose to calculate weights w_i on the basis of a certain probability $P_n^{(i)}$, which we define as a probability that a node i has received n update contacts that were not received by any of the other nodes. We calculate this probability using combinatorics, assuming that each combination of the contact matrix (see Table 1) has equal probability, thus obtaining:

$$P_n^{(i)} = \frac{\binom{K}{k_i} \sum_{l=n}^{k_i} \left[(-1)^{l-n} \binom{k_i}{l} \left(\prod_{j=1, \dots, N}^{j \neq i} \binom{K-l}{k_j} - Z_l^{(i)} \right) \right]}{\prod_{j=1, \dots, N} \binom{K}{k_j} - Z} \quad (1)$$

Table 1. Contact matrix (left part). Local information (right part).

Nodes	Update Ants					Weights	Info
	1	2	3	...	K		
Node 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,j}$	$k_{1,K}$	w_1	m_1
...				$k_{i,j}$		w_i	m_i
Node N	$k_{N,1}$				$k_{N,K}$	w_N	m_N

where $Z_l^{(i)}$ and Z exclude the columns containing only zeros in the contact matrix and are calculated as follows: $Z_l^{(i)} = \sum_{z=1}^{K-k_i} \left[(-1)^{z-1} \binom{K-k_i}{z} \prod_{j=1, \dots, N}^{j \neq i} \binom{K-z-l}{k_j} \right]$ and $Z = \sum_{z=1}^{K-k_{max}} \left[(-1)^{z-1} \binom{K}{z} \prod_{j=1, \dots, N} \binom{K-z}{k_j} \right]$, where k_{max} is the maximum number of contacts received in any node.

We propose to calculate the weight w_i as the expected value of the number of contacts received only by the node i (not received by any other node), namely: $w_i = \sum_{n=1}^{k_i} n P_n^{(i)}$.

For the complete scheme of calculating the synchronized information, we need only the total number of contacts K that occurred in the given location after the last synchronization process. We may estimate it on the basis of known k_i , assuming the Bernoulli distribution of a number of received contacts.

The implementation of the mechanism is as follows. Update contacts are obtained through the MAC layer. When a synchronization is needed, a node requests the information by a broadcast message. Other nodes respond by sending their information together with the number of update contacts received since the last synchronization. The requesting node calculates synchronization weights, synchronizes the knowledge and broadcasts the synchronized knowledge. All the nodes switch to the new version of knowledge and reset their contacts memory.

4 Experimental Results

We performed our experiments using the NS2 network simulator. We compared AntHocNet and AntHocGeo with AODV algorithm [10]. We also compared the “abstract nodes” implementation of AntHocGeo [6,7] (AntHocGeo_A) with the implementation using distributed knowledge exchange mechanism (AntHocGeo).

In a typical ad-hoc scenario we simulate situating 30 nodes on a plane of size 1000m by 1000m. In the first experiment (Fig. 1, top), we study the influence of network’s dynamics on the network’s performance by varying the nodes speed. One may see that using the knowledge exchange mechanism shows the advantage of AntHocGeo over the competitors in terms of delay. As expected, the delay is slightly bigger than in the case of “abstract nodes” implementation, according to the additional overhead. We studied the influence of the network connectivity on our routing algorithm’s performance in the second experiment (Fig. 1, bottom). For the number of nodes lower than 80, we observe the similar results as in the previous experiment: AntHocGeo’s delay is slightly bigger and delivery ratio is higher in comparison to AntHocGeo_A. However, things get worse for higher number of nodes. This may indicate that the overhead of the knowledge exchange mechanism increases with the number of nodes and require additional mechanism for its reduction.

We performed the similar experiments in the network of mobile robots and compare the results with [6]. Robots move across the map of indoor rooms. The geographical localization of knowledge proved to be very effective in the considered scenario. In both experiments (Fig. 2), the results of AntHocGeo

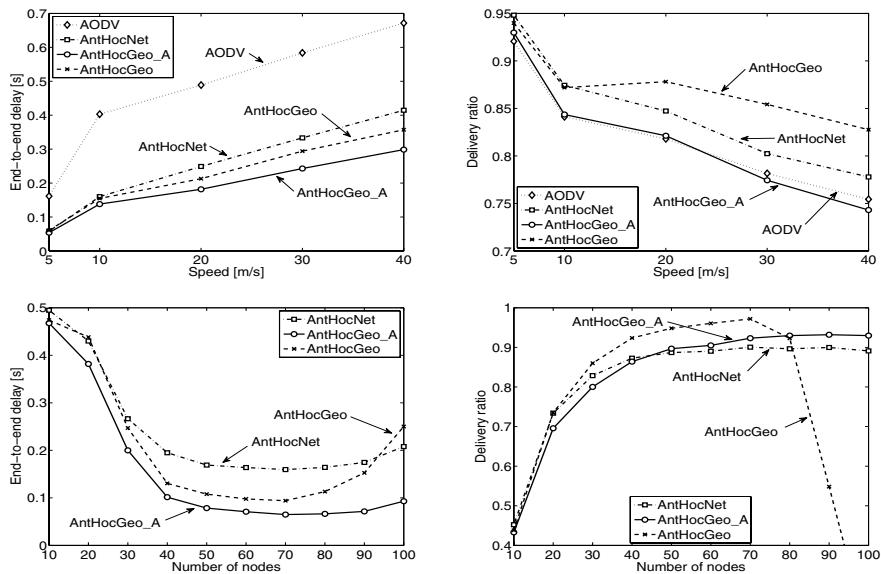


Fig. 1. The average packet delay (upper left) and the delivery ratio (upper right) vs. node speed. The average packet delay (lower left) and the delivery ratio (lower right) vs. node density. Typical ad-hoc scenario.

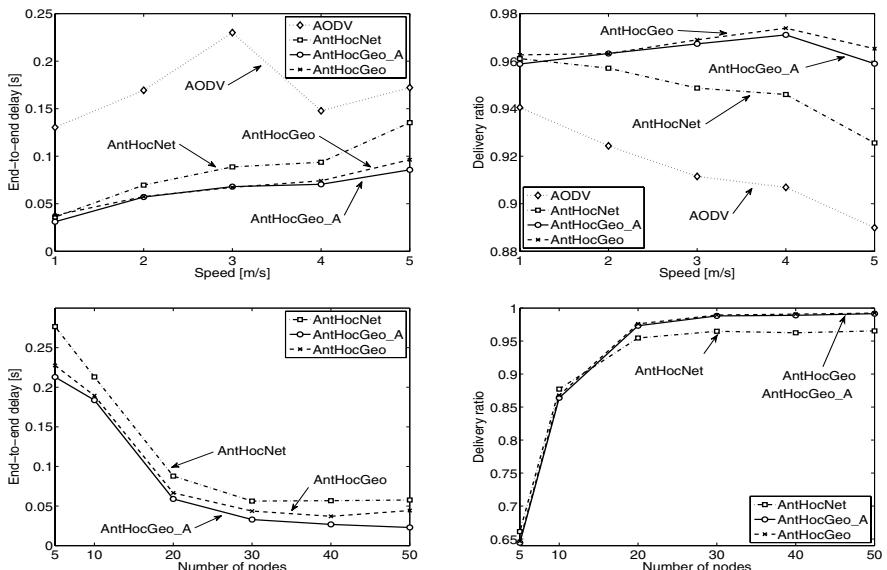


Fig. 2. Same as in Fig. 1. Ad-hoc network of mobile robots

are almost as good as the ones obtained by the implementation with “abstract nodes”. Both implementations show superiority in terms of delay and delivery ratio as compared to AODV and AntHocNet.

5 Conclusions

We proposed and investigated an innovative knowledge management scheme for ant routing in ad-hoc networks. The introduced knowledge exchange mechanism provides the practical realization of the geographical localization of knowledge concept. Simulation experiments performed for various communication scenarios show that the proposed idea may improve the performance of the underlying routing algorithm in terms of delay and delivery ratio. Good behavior of the proposed distributed geographical localization of the information, which replaced the “abstract nodes” considered in previous works, showed that the proposed mechanism can be applied in practice.

It is worth noticing that the proposed solution is not a geographical routing protocol but rather a new way of organizing the routing information. Nevertheless, it is possible to apply it together with geographical routing.

Acknowledgments

Work supported by Warsaw University of Technology Research Program grant.

References

1. Abolhasan, M., Wysocki, T., Dutkiewicz, E.: A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks* 2, 1–22 (2004)
2. Di Caro, G., Ducatelle, F., Gambardella, L.M.: AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks. *European Transactions on Telecommunications* 16(5), 443–455 (2005)
3. Hong, X., Xu, K., Gerla, M.: Scalable Routing Protocols for Mobile Ad Hoc Networks. *Network* 16, 11–21 (2002)
4. Karp, B., Kung, H.T.: Greedy perimeter stateless routing for wireless networks. In: ACM/IEEE MobiCom, pp. 243–254 (2000)
5. Ko, Y.B., Vaidya, N.H.: Location-aided routing (LAR) in mobile ad hoc networks. In: ACM/IEEE MobiCom, pp. 66–75 (1998)
6. Kudelski, M., Gadomska-Kudelska, M., Pacut, A.: Geographical Cells Routing in Ad-Hoc Networks of Mobile Robots. In: IEEE MELECON, pp. 374–379 (2008)
7. Kudelski, M., Pacut, A.: Geographical Cells: Location-Aware Adaptive Routing Scheme for Ad-Hoc Networks. In: IEEE EUROCON, pp. 649–656 (2007)
8. Kudelski, M., Pacut, A.: Learning methods in ad-hoc networks: a review. *Evolutionary Computation and Global Optimization*, Prace Naukowe Politechniki Warszawskiej, Elektronika z 160, 153–163 (2007)
9. Liao, W., Sheu, J., Tseng, Y.: GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks. *Telecommunication Systems* 18, 37–60 (2004)
10. Perkins, C., Royer, E.M.: Ad hoc On-Demand Distance Vector Routing. In: 2nd IEEE WMCSA, pp. 90–100 (1999)

Discrete Particle Swarm Optimization for Multiple Destination Routing Problems*

Zhi-hui Zhan and Jun Zhang**

Department of Computer Science, Sun Yat-sen university, China, 510275
junzhang@ieee.org

Abstract. This paper proposes a discrete particle swarm optimization (DPSO) to solve the multiple destination routing (MDR) problems. The problem has been proven to be NP-complete and the traditional heuristics (e.g., the SPH, DNH and ADH) are inefficient in solving it. The particle swarm optimization (PSO) is an efficient global search algorithm and is promising in dealing with complex problems. This paper extends the PSO to a discrete PSO and uses the DPSO to solve the MDR problem. The global search ability and fast convergence ability of the DPSO make it efficient to the problem. Experiments based on the benchmarks from the OR-library show that the DPSO obtains better results when compared with traditional heuristic algorithms, and also outperforms the GA-based algorithm with faster convergence speed.

Keywords: Particle swarm optimization (PSO), multiple destination routing problems, Steiner tree problem.

1 Introduction

Multiple destination routing (MDR) problems have drawn lots of attentions all over the world in recent years as the fast developments of variants of networks [1]. The problem can be reduced to the Steiner tree problem (STP) in graph. Suppose $N = \{V, E\}$ is an undirected, connected, cost network where V is the nodes set and E is the edges set. A positive function c is used to evaluate the cost of each edge e in E . A subset D of the V contains the destination nodes (the source node is also regarded as a destination node), and a subset $S = V/D$ denotes the intermediate nodes set. The objective of the MDR is to find out a minimal cost tree T that connects all the destination nodes and some of the relevant intermediate nodes. If T is denoted as $T = \{V^*, E^*\}$, then the objective of the T can be formulated as (1):

$$f = \text{Min}T = \min \sum_{e \in E^*} c(e), \text{where } V^* \subseteq V, E^* \subseteq E, D \subseteq V^*. \quad (1)$$

* This work was supported by NSFC Joint Fund with Guangdong, Key Project No. U0835002, NSF of China Project No.60573066 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, P.R. China.

** Corresponding author.

The problem has been proven to be NP-complete [1]. Therefore it is attractive to design heuristic algorithms to deal with this problem, such as the shortest path heuristic (SPH), the distance network heuristic (DNH) and the average distance heuristic (ADH) [2]. These heuristics can yield good performance sometimes, but have their inherent weakness of being trapped into local optima [1].

With the developments of the evolutionary computation (EC), the genetic algorithms (GAs) have been successfully designed to dealing with the MDR problems [1]. However, the GA-based algorithms may encounter the long computational time to obtain the solutions because they have to iterate many generations to converge.

Recently, a new EC paradigm named particle swarm optimization (PSO) has been developed [3]. The PSO emulates the swarm behaviors of birds flocking and fish schooling. While optimizing a problem in a N -dimension hyperspace, each particle i keeps a velocity vector $\mathbf{V}_i = [v_{i1}, v_{i2}, \dots, v_{iN}]$ and a position vector $\mathbf{X}_i = [x_{i1}, x_{i2}, \dots, x_{iN}]$ to indicate the current status. Moreover, the particle i will keep its personal historical best position vector $\mathbf{P}_i = [p_{i1}, p_{i2}, \dots, p_{iN}]$, and the best position of all the \mathbf{P}_i in the whole population is denoted as the globally best \mathbf{G} . The \mathbf{V}_i and \mathbf{X}_i are initialized randomly and are updated as (2) and (3) generation by generation by the guidance of \mathbf{P}_i and \mathbf{G} .

$$v_{id} = \omega \times v_{id} + c_1 \times r_{1d} \times (p_{id} - x_{id}) + c_2 \times r_{2d} \times (g_d - x_{id}). \quad (2)$$

$$x_i^d = x_i^d + v_i^d. \quad (3)$$

The original PSO is also known as global version PSO (GPSO) that each particle learns from its own best position and the globally best position. This PSO version has very fast convergence speed to reasonable solutions, but this may cause the local optima. Researches have been conducted to investigate the affections of different topological structures on the algorithm performance and a local version PSO (LPSO) was proposed [4] to deal with the complex problems.

When compared with other EC algorithms, the PSO is simpler in concept, easier to implement, and with much faster convergence speed [3]. Therefore, we investigate the ability and the advantages of the PSO in solving the MDR problems in this paper. Recently, our group has proposed a novel discrete PSO (DPSO) to solve the STP and resulted in very promising results when compared with some other heuristics and the GA [5]. However, the DPSO in [5] works well by additionally introducing a new parameter and by using a mutation operator to avoid the premature convergence. The mutation, however, is not a natural operator in the PSO algorithm. Therefore, the DPSO in [5] is somehow not a very natural way to use the PSO algorithm. Differently, this paper will focus on designing an effective and efficient DPSO to solve the MDR problems by using topological structure to avoiding the prematurity, but not with a mutation like operation.

The rest of this paper is organized as follows. Section 2 proposes the DPSO for the MDR problems. Section 3 compares the DPSO with some other heuristics and the GA. Conclusions are drawn in Section 4.

2 Discrete PSO for MDR Problems

2.1 Preprocess: Cost Complete Graph

Before using the DPSO to optimize the MDR problem, a pre-process is firstly executed to transform the connected network into a cost complete graph, i.e., find out the shortest path that connects any two different nodes. One of the efficient algorithms for this pre-process is the Floyd's algorithm [6].

In the cost complete graph, every two nodes are connected by a shortest path. However, it is should be noticed that some of the shortest paths are “real” paths means that the path exists in the original network and connects the two nodes directly, while some may be “virtual” paths means that the path does not directly connect the two nodes but via some other intermediate nodes.

2.2 Particle Representation

The particle is coded with a binary string. Here, the length of the string is the same as the total nodes N in the network, and all the destination nodes are with a bit 1 to indicate that these nodes are always in the tree. Moreover, if an intermediate node is with the position 1, it means that the node is selected as the relevant intermediate nodes and is used to construct the tree. Otherwise, this node is not used to construct the MDR tree. Therefore, the position representation of each particle i is as (4):

$$\mathbf{X}_i = [x_{i1}, x_{i2}, \dots, x_{iN}], \text{ where } x_{ij} = 0 \text{ or } 1. \quad (4)$$

while the velocity is represented as (5)

$$\mathbf{V}_i = \begin{bmatrix} v_{i1}^0, v_{i2}^0, \dots, v_{iN}^0 \\ v_{i1}^1, v_{i2}^1, \dots, v_{iN}^1 \end{bmatrix} \text{ where } 0 \leq v_{ij}^0, v_{ij}^1 \leq 1. \quad (5)$$

In (5), the j^{th} dimension of the vector \mathbf{V}_i is associated with two values. The v_{ij}^0 and the v_{ij}^1 are the probabilities that the x_{ij} to be 0 and to be 1, respectively. It should be noticed that the sum of v_{ij}^0 and v_{ij}^1 is unnecessary to be 1.

2.3 Velocity and Position Update

The following modifications are made to the velocity and position update processes.

1) Position-Position: The velocity update involves the term “Position-Position”. Suppose that the two positions are \mathbf{X}_1 and \mathbf{X}_2 , and $\mathbf{V} = \mathbf{X}_1 - \mathbf{X}_2$. For the j^{th} dimension of the \mathbf{V} , if x_{1j} is b but x_{2j} is not b (b is 0 or 1), then the particle should learn from the \mathbf{X}_1 , hence $v_j^b=1$; if x_{1j} is the same as x_{2j} and is b , it means that it is unnecessary for the \mathbf{X}_2 to learn from the \mathbf{X}_1 on this corresponding dimension, hence $v_j^b=0$.

2) Coefficient \times Velocity: The coefficient can be the inertia weight ω , or the acceleration coefficient $c \times r$. The operation of “Coefficient \times Velocity” is to

multiply the coefficient with each element in the velocity. If the production is larger than 1, the value is set to 1, making sure that the range is within [0, 1].

3) Velocity+Velocity: The result of “Velocity+Velocity” is a new velocity. Suppose that $\mathbf{V} = \mathbf{V}_1 + \mathbf{V}_2$, the j^{th} dimension in \mathbf{V} is the same as the larger one between v_{1j}^b and v_{2j}^b , where $b=0, 1$.

4) Position Update: For the j^{th} dimension of the \mathbf{X}_i , a random value α in [0, 1] is generated firstly. If both the v_{ij}^0 and v_{ij}^1 are larger than α , then x_{ij} is set to 0 or 1 randomly. If only the v_{ij}^b is larger than α , then x_{ij} is set to b , where b is 0 or 1. If both of v_{ij}^0 and v_{ij}^1 are smaller than α , then x_{ij} is set as itself.

2.4 Evaluation: Minimal Spanning Tree and Prune

The construction the multicast tree is based on the modified Prim’s minimal spanning tree (MST) algorithm [7] that is described as follows.

We firstly select a random destination node (the source node is also regarded as a destination node) into T , and then select the other used nodes (the node with bit 1 in the position \mathbf{X}) one by one based on their distances to the T . During the selecting, we prefer the real path to the virtual path even though the real cost is larger than the virtual cost. However, if there are no nodes connect the T with real path, we prefer the shortest virtual path. When a virtual path is selected, the corresponding intermediate nodes are also added into the tree. This constructing process determines when all the destination nodes have been selected in to the T .

The constructed multicast tree may have some redundant nodes. Therefore, we delete all the leaves in the T that connect the non-destination nodes. We find out the node i in T that has the degree of 1, if the node i is not a destination node, delete it and reduce the degree of the node j that connects i by 1. Repeat doing this until no non-destination nodes with degree 1 can be found.

During the above processes, the cost of the multicast tree can be calculated. Therefore, the fitness of the particle can be evaluated.

3 Experiments and Comparisons

3.1 Problems and Algorithms

The experiments are based on the STPs in graphs from Category B in the OR-library (<http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/> [steinb1.txt to steinb18.txt]). Moreover, the DPSOs are based on the global version PSO (GPSO) and the local version PSO (LPSO), and they are named D-GPSO and D-LPSO respectively. In the DPSOs, the population size M is set to 20, and the inertia weight ω is linearly decreasing from 0.9 to 0.4. The acceleration coefficients c_1 and c_2 are both set to 2.0.

For all the DPSOs, the maximal generation is 1250, and therefore there are at most $20 \times 1250 = 25000$ function evaluations (FEs) each run, the same as the configurations used in [5]. The algorithm terminates when it finds the global

optimum or it reaches the maximal generation. Each problem was simulated for 100 times independently.

3.2 Experimental Results and Comparisons

The solution accuracy and the convergence speed of the experimental results obtained by different algorithms are compared in Table 1. The $|V|$, $|E|$ and $|D|$ denote the nodes number, edges number and destination nodes number of the MDR problem, respectively. The OPT is the optimal solution provided by the OR-library. The term R is the relative error of the result defined as $R = (Result - OPT)/OPT$. The R values of the traditional heuristics SPH, DNH, ADH, and the GA have been reported in [1]. The FEs denotes the mean function evaluations to obtain the OPT . The FEs values for the GA are reported in [5] and are used here for comparisons.

Table 1. Comparisons on solution accuracy and convergence speed

Problem No.	Details				Solution $R(\%)$ and FEs, $R = (Result - OPT)/OPT$					
	$ V $	$ E $	$ D $	OPT	SPH	DNH	ADH	GA	D-GPSO	D-LPSO
B01	50	63	9	82	0	0	0	0 (105)*	0 (43)*	0 (41.6)*
B02	50	63	13	83	0	8.43	0	0 (160)	0 (55)	0 (56.6)
B03	50	63	25	138	0	1.45	0	0 (100)	0 (45.6)	0 (45.8)
B04	50	100	9	59	5.08	8.47	5.08	0 (120)	0 (75.2)	0 (89.6)
B05	50	100	13	61	0	4.92	0	0 (130)	0 (64.4)	0 (67)
B06	50	100	25	122	0	4.92	0	0 (515)	0.52 (377.4)	0 (227.4)
B07	75	94	13	111	0	0	0	0 (275)	0 (45.4)	0 (45)
B08	75	94	19	104	0	0	0	0 (185)	0 (46.4)	0 (47.8)
B09	75	94	38	220	0	2.27	0	0 (275)	0 (56.6)	0 (62.2)
B10	75	150	13	86	4.65	13.95	4.65	0 (780)	0 (82.4)	0 (104)
B11	75	150	19	88	2.27	2.27	2.27	0 (585)	0.02 (91.1)	0 (104.2)
B12	75	150	38	174	0	0	0	0 (260)	0 (131.6)	0 (169.4)
B13	100	125	17	165	7.88	6.06	4.24	0 (1100)	0.12 (86)	0 (104.6)
B14	100	125	25	225	2.55	1.28	0.43	0 (4020)	0 (207.2)	0 (105.8)
B15	100	125	50	318	0	2.20	0	0 (1380)	0 (81.2)	0 (95.4)
B16	100	200	17	127	3.15	7.87	0	0 (885)	0.39 (110.3)	0 (147.4)
B17	100	200	25	131	3.82	5.34	3.05	0 (925)	0 (178)	0 (255.8)
B18	100	200	50	218	1.83	4.59	0	0 (1250)	0.01 (218.2)	0 (332.4)

* The numbers are the mean function evaluations (FEs) to obtain the OPT

The comparisons show that the traditional heuristics are easy to be trapped into local optima and result in poor solution accuracy. The relative errors for the SPH, DNH and ADH on some of the problems are even larger than 5%, indicating that these algorithms may not be promising in producing high accuracy solutions. The GA can obtain the OPT on all the problems in every run and all the relative errors are 0%. The D-LPSO also has the good performance to obtain the OPT for all the problems in every run. Even though the D-GPSO can not succeed in obtaining the OPT on problems B06, B11, B13, B16 and

B18, the relative error values are very small, showing that the algorithm is still very promising in producing good solutions.

The comparisons also show that our proposed DPSOs have a much faster convergence speed than the GA. Both the D-GPSO and the D-LPSO can obtain the global optimal *OPT* with very fewer FEs than the GA. When compare the D-GPSO and the D-LPSO, it can be observed that the D-GPSO is always faster than the D-LPSO. This phenomenon, however, is consistent with the one in continuous domain that the global version PSO always converges faster than the local version PSO [4]. However, the LPSO may be superior to the GPSO when dealing with complex multimodal problems and can result in better final solutions [4]. Similarly, in our experiments, the solutions obtained by the D-LPSO are indeed better than those obtained by the D-GPSO while the convergence speed of the D-GPSO is faster than the D-LPSO.

4 Conclusions

A discrete particle swarm optimization (DPSO) has been proposed in this paper to solve the MDR problems. The DPSO uses the binary code to represent the position and introduces a new representation for the velocity. Modifications have been made to the velocity and position update processes in order to keep the learning concept of the original PSO in continuous domain. Experimental results show that the DPSO can obtain better solutions than the traditional heuristics and can converge much faster than the GA-based algorithm in term of function evaluations.

References

1. Leung, Y., Li, G., Xu, Z.B.: A genetic algorithm for the multiple destination routing problems. *J. IEEE Trans. Evol. Comput.* 2, 150–161 (1998)
2. Winter, P., Smith, J.M.: Path-Distance heuristics for the Steiner problem in undirected networks. *J. Algorithmica* 7, 309–327 (1992)
3. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the IEEE Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
4. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, HI, pp. 1671–1676 (1995)
5. Zhong, W.L., Huang, J., Zhang, J.: A novel particle swarm optimization for the Steiner tree problem in graphs. In: Proceedings of the IEEE Congress on Evolutionary Computation, Hong Kong, pp. 2465–2472 (2008)
6. Floyd, R.W.: Shortest path. *J. Communications of the ACM* 5, 345 (1962)
7. Prim, R.C.: Shortest connection networks and some generalizations. *J. Bell System Technical Journal* 36, 1389–1401 (1957)

Combining Back-Propagation and Genetic Algorithms to Train Neural Networks for Ambient Temperature Modeling in Italy

Francesco Ceravolo¹, Matteo De Felice^{1,2}, and Stefano Pizzuti¹

¹ Energy, New technology and Environment Agency (ENEA), Via Anguillarese 301,
00123 Rome, Italy

{francesco.ceravolo,matteo.defelice,stefano.pizzuti}@enea.it

² University of Rome “Roma 3” Department of Informatics and Automation, Via
della Vasca Navale 79, 00146 Rome, Italy

Abstract. This paper presents a hybrid approach based on soft computing techniques in order to estimate ambient temperature for those places where such datum is not available. Indeed, we combine the Back-Propagation (BP) algorithm and the Simple Genetic Algorithm (GA) in order to effectively train neural networks in such a way that the BP algorithm initialises a few individuals of the GA’s population. Experiments have been performed over all the available Italian places and results have shown a remarkable improvement in accuracy compared to the single and traditional methods.

Keywords: Neural Networks, Back-Propagation Algorithm, Simple Genetic Algorithm, Ambient Temperature Modeling, Sustainable Building Design.

1 Introduction

One of the most important issues in environmental protection is the one related to energy consumption and pollutant emissions associated to buildings, indeed in recent years some studies have been carried out in the area of the so called “sustainable buildings” [3,8]. Sustainable buildings have a remarkable impact on environment because they use energy in a more effective way which turns out into less CO₂ emissions for heating, less primary energy consumption and better thermal comfort.

In this context, the design phase is very critical and depends on simulations where the accuracy estimation of several environmental parameters is crucial. Among these, the most important ones are: solar radiation, ambient temperature and ambient humidity. The first one is relatively simple to be modeled and some work has already been done [10,14,23]. The others are tougher and very little has been carried out. Therefore, in this paper we tackle the problem of ambient temperature modeling.

When approaching this task two principal problems rise: the only available data are often based on monthly averages; the existing data regard few places

generally nearby airports. As concerns the first problem, algorithms which estimate reliable hourly values given the monthly ones are already existing. [4] The second problem is slightly tougher. So far, tools like TRNSYS [12] are based on the fact that data are taken from historical databases and if it occurs that the database is lacking in the information we are looking for , the data of the nearest unknown location will be given as an estimate of the unknown one (the Nearest Neighbor algorithm - NN). This approach does not satisfy since climate is a highly non linear system and depends on a large number of variables. Indeed, it is not always true that locations close to each other have similar environmental behaviors (for example temperature profiles) and it is the cause of large errors.

On the other hand, the scientific world has shown an increased interest in soft computing methodologies, like Artificial Neural Networks (ANNs) [1,7], which have proved [13,19] to be powerful tools to solve complex modeling problems for non-linear systems. Indeed, some applications of soft computing techniques in the field of modeling environmental parameters[17,22,24,25] have provided interesting results. Back-Propagation (BP)[20] and Genetic Algorithms (GA)[6,9] are among the most used algorithms to train ANNs. Despite the success of the algorithms, each has its own drawback [5,16]. As a deterministic gradient-descent algorithm and stochastic technique respectively, there might exist a balance between their advantages and disadvantages. Among the possible combinations of the two methods the most interesting ones are basically three: the first is the one which uses the BP as initialization for the GA (BPGA)[15,26], the second uses the GA as initialization for the BP (GABP) [11] and the third one uses the GA for best training set generation for the BP [21].

In this context, we carried out a BPGA method in order to train ANNs as ambient temperature models. What is new in this work is that we train several ANNs with BP to initialize several individuals of the GA population, rather than only one as reported in the cited BPGA methods. Moreover, the application of the method is innovative and its main advantage is that we have a non linear interpolation tool capable to provide a reliable temperature estimate when dealing with places not present in the database. Experimentation has been performed on all the available Italian localities.

2 The Methodologies

In this work we applied the following two methodologies: Nearest Neighbor (NN), and the BPGA method.

2.1 Nearest Neighbor

This is a very simple, and for this reason widely used, method to estimate unknown climate data of a locality because it provides as estimation the data of the closest known locality according to the following formula:

$$t = t_i \quad (1)$$

Where t is the parameter to be estimated (for instance ambient temperature) and t_i is the datum of the i^{th} locality which corresponds to the closest one, therefore

$$i = \min(d_j) \quad (2)$$

for $j = 1, \dots, n$ where n is the number of known localities and

$$d_j = \sqrt{(x - x_j)^2 + (y - y_j)^2} \quad (3)$$

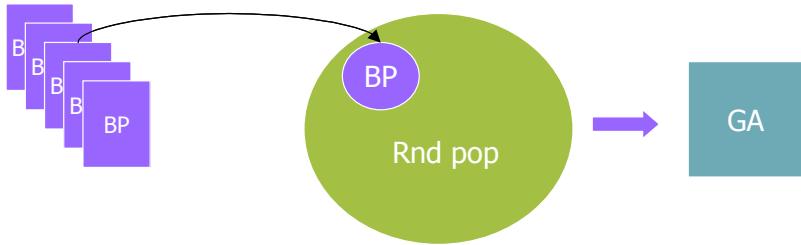
where x and y are the latitude and longitude of the requested locality and x_j and y_j are the latitude and longitude of the known localities.

2.2 The BP-GA Combination

The BP is a gradient-descent algorithm used to find the neural network weights so that the estimate error function reaches the minimum. It can be proved that BP can reach the extremum within a limited number of epochs. The merits of BP are that the adjustment of weights is always towards the descending direction of the error function and that only some local information is needed. On the other hand, BP also has its disadvantages. For example, the error curve is generally so complex that there are a lot of local minima making the convergence of the algorithm very sensitive to the initial values. GAs are parallel stochastic optimization algorithms. As compared with BP, GAs are more qualified for neural networks if only the requirement of a global searching is considered. However, the price one pays for GAs is the slowness which is mainly due to the random initialization of the genes and to the slow but crucial exploration mechanisms employed, which has three basic arithmetic operators: reproduction, crossover and mutation. Another shortcoming of GA is that the method is not theoretically perfect and it cannot ensure convergence and achievement of the optimum. From the analysis above, it is easy to observe the complementarity between BP and GA. The proposed BPGA can learn from their strong points to offset their weakness. It can be typically applied to MLP (Multi-Layer Perceptron) supposed that the activation functions of the hidden layers and the output layer are all sigmoidal.

In the proposed BPGA, BP is first used to train several neural networks (a small fraction of the total GA's population size) for approximately 1000000 cycles with no early stopping criterion. Then, the weights of the BP computations are encoded into several chromosomes of the GA's initial population together with other randomly generated chromosomes (Fig.1).

From this initial population, subsequent populations are being computed for a small number of performance requests (about 10000). In this stage, in order to avoid over-fitting the “save best” approach is applied. The GA we implemented is Holland's Simple Genetic Algorithm[9] with 1-elitism. The save best method has been implemented using a small part of the training set as testing set. The individuals are trained on the training set but the champion of the population is periodically run over the testing set and if the current champion has a lower

**Fig. 1.** BPGA method

testing error than the previous champion, then the current error will be the new best and the champion chromosome will be saved. At the end of the training stage, the best saved state is adopted to be the used one for the validation stage.

The GA and BPGA parameters are reported in tab.1

Table 1. Parameter settings

	Population size	Chromosomes BP initialized	Crossover rate	Mutation rate	Performance requests
GA	100	0	0.9	0.1	10^6
BPGA	100	5	0.9(GA)	0.1(GA)	10^6 (BP) + $+10^4$ (GA)

Therefore, in the proposed method the main advantage is that the searching domain of the GAs is reduced and thus, the convergence time is shortened. Moreover, the feature of parallel optimization of GAs may help the BP networks getting out of the local minima which they tend to plunge into.

3 Experimentation

Experimentation concerned the comparison of 4 different methods for ambient temperature modeling: the Nearest Neighbor algorithm (NN), the simple genetic algorithm (GA), the Back-Propagation algorithm and the proposed BPGA method. The last three were applied to train neural networks. Tests have been carried out over all the data available for Italian localities [2]. The whole data set (740 localities) has been split into nine smaller sets according to climate areas and then each of these has been partitioned into training and validation sets (Tab. 2). Each locality has twelve different records, each corresponding to the monthly average temperature.

As regards neural networks, the architecture we applied is a simple feed-forward one with 4 input neurons (latitude, longitude, height above sea level and the month of the year), 6 to 10 hidden neurons and one output neuron (the average monthly temperature). All the transfer functions are the classic sigmoid function. The implemented BP method has constant learning rate (1.0) and adaptive self-momentum.

Table 2. Data set partitioning

	Regions	Training set size (localities)	Validation set size (localities)
Area1	V.dAosta, Piemonte, Lombardia	76	17
Area2	TrentinoAA, Veneto, FriuliVG, Emilia-Romagna	161	18
Area3	Liguria, Toscana, Umbria	90	11
Area4	Marche, Abruzzo	43	13
Area5	Lazio, Campania	71	9
Area6	Puglia, Molise	66	8
Area7	Basilicata, Calabria	53	6
Area8	Sardegna	36	5
Area9	Sicilia	47	10
TOTAL	20	643	97

Table 3. Validation results: average absolute error (C)

	NN	GA	BP	BPGA
Area1	1.07	1.16(± 0.05)	0.80(± 0.15)	0.66(± 0.01)
Area2	1.47	1.12(± 0.13)	0.86(± 0.18)	0.66(± 0.02)
Area3	0.93	0.81(± 0.07)	1.12(± 0.35)	0.70(± 0.03)
Area4	1.0	1.98(± 0.08)	0.79(± 0.04)	0.66(± 0.01)
Area5	1.59	0.77(± 0.10)	0.60(± 0.02)	0.53(± 0.02)
Area6	1.28	0.85(± 0.13)	0.72(± 0.07)	0.64(± 0.01)
Area7	1.28	1.06(± 0.08)	0.75(± 0.20)	0.65(± 0.04)
Area8	0.65	0.84(± 0.10)	0.99(± 0.39)	0.54(± 0.03)
Area9	3.11	1.88(± 0.60)	1.10(± 0.09)	0.50(± 0.02)
Average	1.3	1.12(± 0.15)	0.86(± 0.17)	0.62(± 0.02)

Table 4. Validation results: maximum absolute error (C)

	NN	GA	BP	BPGA
Area1	7.2	3.5(± 0.20)	2.60(± 0.60)	2.70(± 0.02)
Area2	6.8	5.10(± 0.30)	2.48(± 0.65)	2.40(± 0.10)
Area3	3.5	3.66(± 0.10)	2.95(± 0.80)	2.50(± 0.05)
Area4	4.2	5.40(± 0.15)	2.99(± 0.07)	2.77(± 0.02)
Area5	4.4	2.63(± 0.12)	1.47(± 0.13)	1.55(± 0.05)
Area6	4.0	2.73(± 0.11)	2.15(± 0.40)	2.32(± 0.02)
Area7	2.7	2.74(± 0.25)	1.66(± 0.36)	1.65(± 0.15)
Area8	3.0	3.14(± 0.10)	2.24(± 0.34)	1.75(± 0.05)
Area9	9.10	5.10(± 1.50)	2.93(± 0.47)	1.77(± 0.02)
Average	4.99	3.78(± 0.30)	2.39(± 0.42)	2.16(± 0.05)

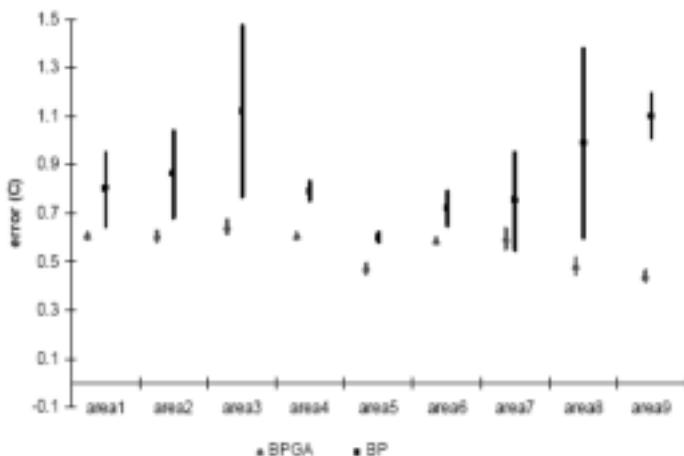


Fig. 2. Comparison of average error and standard deviations

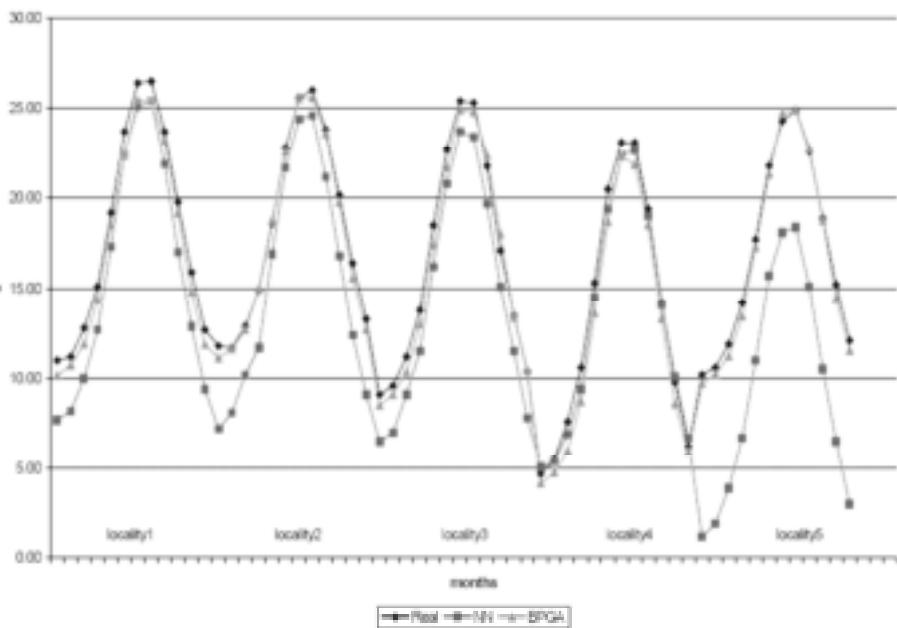


Fig. 3. Comparison of the temperature modeling testing results (area9)

Moreover, in Tab. 3 and Tab. 4 we report the average and maximum absolute errors (with the standard deviations in brackets) averaged over 30 runs obtained on the validation sets.

Experimentation clearly shows the effectiveness of the proposed BPGA approach. In fact, it always outperforms all the other methods in terms of average

and, which is more relevant, maximum absolute estimation error. Furthermore, it is interesting to point out that the BPGA average standard deviation is very little (0.02° C), meaning that the method is robust and reliable. To stress this achievement, in fig.2 we can see a graph representing Tab. 3.

The reasons for this are mainly due to the fact that the searching domain of the GA is cut down by the BP initialization and that the GA's parallel optimization gets the BP out of the local minima which it gets stuck in.

Moreover, it is to consider that such results have been achieved using as input only the minimal information which is always available, which is the geographical coordinates, without taking into account other important environmental parameters, like pressure, humidity and wind, which are accessible only for few localities (mainly those with an airport).

Finally, as example we report a graph (fig.3) comparing the real monthly temperature to the one estimated by the nearest neighbor (NN) method and by the proposed approach (BPGA) over five localities belonging to area 9.

4 Conclusion

In this paper we tackled the issue of ambient temperature modeling since it is one of the most important environmental parameters when designing effective sustainable buildings. To solve this problem we proposed a hybrid approach based on soft computing techniques in order to provide ambient temperature for those places where such data are not available. Indeed, we combined the Back-Propagation algorithm and the Simple Genetic Algorithm (BPGA) in order to effectively train neural networks in such a way that the BP algorithm initializes a few individuals of the GA's initial population.

Experiments were performed over all the available Italian localities and results showed a remarkable improvement in accuracy compared to the single (BP and GA) and traditional methods (the Nearest Neighbor algorithm - NN). In particular, with respect to the NN approach (the most used one in commercial software) the average modeling error is halved (from 1.3° C to 0.62° C) and the maximum one is reduced of one third in the worst case (from 9° C to 2.8° C). Moreover, The BPGA method showed very high robustness and reliability (very low standard deviations).

The reason for this success is due to the fact that the BPGA algorithm combines BP and GA in such a way that the virtues of the single methods are enhanced. Indeed, the BP is first applied so that the searching domain of GA is trimmed down, reducing therefore the GA convergence time, and then the parallel GA optimization extricates the BP from the local minima which it plunges into.

Therefore, the main advantage of this method is that we have a non linear interpolation tool capable to provide a reliable temperature estimate when dealing with unknown places, and this is a critical point to effectively design sustainable buildings.

Future work will focus on the experimentation with different techniques like Radial Basis Function Networks (RBF) and Support Vector Machines (SVM)

as well as different evolutionary algorithms and ensembling methods. Moreover, the proposed technique can be applied to model other complex environmental parameters like ambient humidity.

References

1. Arbib, M.A.: *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge (1995)
2. Atlante Italiano della Radiazione Solare, <http://www.solaritaly.enea.it>
3. Buvik, K., Hestnes, A.G.: Interdisciplinary approach to sustainable building. Experiences from working with a Norwegian demonstration building on retrofitting. *Nordic Journal of Architectural Research* 20(3) (2008)
4. Erbs, D.G., Klein, S.A., Beckman, W.A.: Estimation of degree-days and ambient temperature bin data from monthly-average temperatures. *Ashrae Journal*, 60–65 (1983)
5. Farag, W.A., Quintana, V.H., Lambert-Torres, G.: Genetic Algorithms and Back-Propagation: a Comparative Study. In: Canadian Conference on Electrical and Computer Engineering Proceedings of the 1998 11th Canadian Conference on Electrical and Computer Engineering, CCECE. Part 1, pp. 93–96 (1998)
6. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Reading (1989)
7. Haykin, S.: *Neural Networks, a comprehensive foundation*, 2nd edn. Prentice Hall, New Jersey (1999)
8. Hestnes, A.G.: Integrated design processes - a must for low energy buildings. In: *The Future of Civil Engineering* (2008)
9. Holland, J.H.: *Adaptation in natural and artificial systems*. MIT Press, Cambridge (1975)
10. Jiang, Y.: Prediction of monthly mean daily diffuse solar radiation using artificial neural networks and comparison with other empirical models. *Energy Policy* 36, 3833–3837 (2008)
11. Khan, A.U., Bandopadhyaya, T.K., Sharma, S.: Genetic Algorithm Based Back-propagation Neural Network Performs better than Backpropagation neural Network in Stock Rates Prediction. *Intl. Journal of Computer Science and Network Security* 8(7), 162–166 (2008)
12. Klein, S.A., Beckman, W.A., Mitchell, J.W., Duffie, J.A., Duffie, N.A., Freeman, T.L., Mitchell, J.C., Braun, J.E., Evans, B.L., Kummer, J.P., Urban, R.E., Fiksel, A., Thornton, J.W., Blair, N.J., Williams, P.M., Bradley, D.E. (eds.): *TRNSYS, a Transient System Simulation Program*. Solar Energy Lab, Univ. of Wisconsin-Madison (2000)
13. Kosko, B.: *Neural networks and fuzzy systems*. Prentice-Hall, Englewood Cliffs (1992)
14. Krishnaiah, T., Srinivasa Rao, S., Madhumurthy, K., Reddy, K.S.: Neural Network Approach for Modelling Global Solar Radiation. *Journal of Applied Sciences Research* 3(10), 1105–1111 (2007)
15. Lu, C., Shi, B.: Hybrid Back-Propagation/Genetic Algorithm for Feedforward Neural Networks. In: *ICSP 2000* (2000)
16. McInerney, M., Dhawan, A.P.: Use of Genetic Algorithms with Back Propagation in Training of Feed-Forward Neural Networks. In: *IEEE International Conference on Neural Networks*, pp. 203–208 (1993)

17. Mitra, A.K., Nath, S.: Forecasting maximum temperatures via fuzzy nearest neighbour model over Delhi. *Appl. Comput. Math.* 6(2), 288–294 (2007)
18. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. *IEEE Trans. Neur. Netw.* 1(1), 4–27 (1990)
19. Nikravesh, M., Farell, A.E., Stanford, T.G.: Model identification of non linear time variant processes via artificial neural network. *Computes and Chemical Engineering* 20(11), 1277–1290 (1996)
20. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986)
21. Sadeq, A.M., Wahdan, A.-M.A., Mahdi, H.M.K.: Genetic Algorithms and its use with back-propagation network. *AIN Shams University Scientific Bulleettin* 35(3), 337–348 (2000)
22. Sen, Z.: Fuzzy algorithm for estimation irradiation from sunshine duration. *Solar Energy* 63(1), 39–49 (1998)
23. Soares, J., Oliveira, A.P., Zlata Bozna, M., Mlakar, P., Escobedo, J.F., Machado, A.J.: Modeling hourly diffuse solar-radiation in the city of So Paulo using a neural-network technique. *Applied energy* 79(2), 201–214 (2004)
24. Sodoudi, S.: Estimation of temperature, precipitation and evaporation with Neuro-Fuzzy method. In: *Workshop of statistical downscaling*, Oslo (2005)
25. Tatli, H., Sen, Z.: A new fuzzy modeling approach for predicting the maximum daily temperature from a time series. *Journal of Engineering and Environmental Science* 23, 173–180 (1999)
26. Zhang, M., Ciesielski, V.: Using Back Propagation Algorithm and Genetic Algorithms to Train and Refine Neural Networks for Object Detection. In: *Computer Science Postgraduate Student Conference*, Melbourne (1998)

Estimating the Concentration of Nitrates in Water Samples Using PSO and VNS Approaches

Pablo López-Espí, Sancho Salcedo-Sanz*, Á.M. Pérez-Bellido,
Emilio G. Ortiz-García, Oscar Alonso-Garrido, and Antonio Portilla-Figueras

Department of Signal Theory and Communications, Universidad de Alcalá,
28871 Alcalá de Henares, Madrid, Spain
sancho.salcedo@uah.es

Abstract. In this paper we present a study of the application of a Particle Swarm Optimization (PSO) and a Variable Neighborhood Search (VNS) algorithms to the estimation of the concentration of nitrates in water. Our study starts from the definition a model for the Ultra-violet spectrophotometry transmittance curves of water samples with nitrate content. This model consists in a mixture of polynomial, Fermi and Gaussian functions. Then, optimization algorithms must be used to obtain the optimal parameters of the model which minimize the distance between the modeled transmittance curves and a measured curve (curve fitting process [1]). This process allows us to separate the modeled transmittance curve in several components, one of them associated to the nitrate concentration, which can be used to estimate such concentration. We test our proposal in several laboratory samples consisting in water with different nitrate content, and then in three real samples measured in different locations around Madrid, Spain. In these last set of samples, different contaminant can be found, and the problem is therefore harder. The PSO and VNS algorithms tested show good performance in determining the nitrate concentration of water samples.

1 Introduction

Nitrates are one of the most common pollutants in ground water. It has been reported that elevated nitrates concentrations in drinking water can cause different health problems in children and adults [2], so the detection and control of these contaminants are very important tasks in which huge research is being carried out. Nitrates can be found in dangerous concentrations in ground water mainly due to agricultural activities: nitrogen is a vital nutrient to enhance plant growth. This fact has motivated the intensive use of nitrogen-based fertilizers to increase the productivity of crops in many regions of the world [3]. The problem is, that in many cases nitrogen-rich fertilizers application exceeds the plants demand, so the nitrogen reaches the groundwater.

* This work has been partially supported by Universidad de Alcalá and Comunidad de Madrid with a grant number: CCG07-UAH/TIC-1894. A. M. Pérez-Bellido is supported by Universidad de Alcalá, under the University F.P.I. grants program.

Different methods have been used to accurately estimate the concentration of nitrates in water, but spectrometry has been maybe the most commonly applied [4]-[8]. The majority of the previous works which use a measurement of absorbtion at one or few wavelength suffer from difficulties caused by the interaction of organic matter or other contaminants which interfere the truly nitrate response. For example, sulfates and ions of iron, lead or mercury present absorbtion at the same range than nitrate and nitrite (about 220-230 nm). The treatment of the water sample with re-actives in order to eliminate interfering substances, or modeling the different interfering substances's response are the main methods used up until now to obtain accurate measures of nitrate/nitrite concentrations.

In this paper we propose a novel method to measure the nitrate concentration in water samples from transmittance measures. We propose to use a model of the transmittance curves of nitrate, optimized with PSO or VNS algorithms to obtain the best parameter values for the model. First, the total transmittance curve of the sample is modeled as the product of a number of individual contaminants, in which any number of contaminants can be included. We consider that each contaminant response can also be modeled with a mix of a Fermi and a Gauss distributions (but the organic matter, which can be described using a second order polynomial, because it does not reach a maximum in the study wavelengths). Then, the parameters of the different individual contaminants must be optimized. This optimization is performed in such a way that the contribution of the individual responses are as close as possible to the total measured response (the quadratic error function between the measured and the optimized curves is used as objective function to be minimized). As we have mentioned before, we have tested two of the most innovative metaheuristics of the last few years: a PSO and a VNS algorithms. The VNS algorithm incorporates an ant scheme as local searcher. The performance of the algorithms in this problem is tested in several water samples, with different number of contaminants.

The rest of the paper is structured as follows: next section introduces the model for the transmittance curves of nitrate, using a mix of a Fermi and Gauss function. Section 3 describes the PSO and VNS algorithms used in the paper in order to optimize the model's parameters. In Section 4 the performance of our proposals is shown by analyzing the nitrates concentration of different transmittance curves, some of them measured in aquifers of Madrid, Spain. Section 5 closes the paper giving some final remarks.

2 A Model for Nitrate Transmittance Curves

It is well known that water contaminants derived from nitrogen (mainly nitrates) absorb energy in the ultra-violet range, with a maximum of absorbtion around 220-230 nm. Therefore, spectral analysis for different concentrations of nitrates in pure water provides pattern transmittance curves, which might be used as template in order to obtain the nitrates content of a sample. The problem is that several contaminants may interfere the spectral response of nitrates, so a

direct comparison of the patterns with real samples may lead to over-estimations of the nitrates concentration. As has been mentioned above, organic matter, sulfates and different ions of iron, lead or mercury present absorption at similar wavelength ranges than nitrates contaminants. In addition, turbidity is also a source of error in these measures. The possibilities to accurately measure the concentration of nitrates are then to use a method to eliminate the different interference contaminants and turbidity influence, or modeling the spectral response of nitrates in such a way that the interference can be corrected using adjusting or statistical methods. This paper is based on the latter approach, by defining a novel model for the transmittance curves.

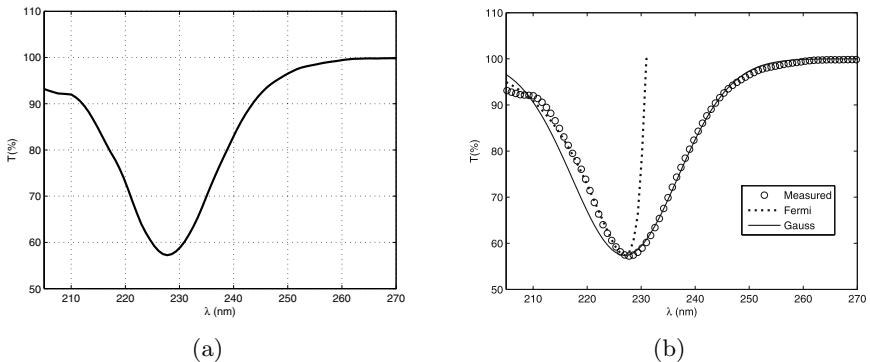


Fig. 1. (a) Measured transmittance of pure water with nitrates (100 ppm); (b) optimized transmittance curve model using the proposed method, both parts of the Fermi and Gauss functions are displayed in this case

The first step of our method consists of accurately modeling the transmittance shape of a nitrate solution in pure water using a set of known functions. As an example consider Figure 1, which shows an example of a measure of the transmittance curve for a concentration of nitrates of 100 ppm. We have chosen a Fermi and a Gauss function to obtain this model. Selection of these two type of functions is not a random election: the processes of spontaneous light emission and absorption in light sources based on semi-conductors follow a Fermi function, whereas light sources are usually modeled using Gauss functions.

The Fermi function has the following expression:

$$f_F(x) = A_1 \cdot \sqrt{x} \cdot e^{-\left(\frac{x}{\sigma_1}\right)} \quad (1)$$

where constants A_1 and σ_1 control the maximum/minimum value and the curve width at half amplitude. On the other hand, a Gauss-type function has the following expression:

$$f_G(x) = A_2 \cdot e^{-\left(\frac{x}{\sigma_2}\right)^2} \quad (2)$$

where constants A_2 and σ_2 control the maximum/minimum value and the curve width at half amplitude.

Note that the shape of the Fermi/Gauss function from their minimum value to the left/right is similar to the shape of the transmittance curves of nitrite, shown in Figure 1. Since the transmittance curve is not symmetric respect to the wavelength of maximum absorbtion (minimum of the transmittance curve), we can model the first part of the transmittance curve, until the minimum, as a Fermi function. The second part of the curve, from the minimum can be modeled as a Gauss function. Thus, the nitrates transmittance curve can be modeled using the following expression:

$$T(\lambda)\% = \begin{cases} 100 \cdot \left(1 - \frac{100-T_{min}(\%)}{100\sqrt{\sigma_1}e^{-0.5}} \sqrt{\lambda_{min} + \frac{\sigma_1}{2}} - \lambda \cdot e^{-\left(\frac{\lambda_{min}+\frac{\sigma_1}{2}-\lambda}{\sigma_1}\right)^2} \right) & \text{if } \lambda \leq \lambda_{min} \\ 100 \cdot \left(1 - \frac{100-T_{min}(\%)}{100} \cdot e^{-\left(\frac{\lambda-\lambda_{min}}{\sigma_1}\right)^2} \right) & \text{if } \lambda > \lambda_{min} \end{cases} \quad (3)$$

Using this model, we can reconstruct the shape of any transmittance curve. Figure 1 (b) shows an example of this reconstruction, depicting the complete Fermi and Gauss functions (the model only considers these functions to or from λ_{min} , see Equation (3)).

The complete model for a real case, in which the water sample contains nitrate and other contaminants such as organic matter, or metallic ions, is the following:

We consider that the transmittance of a real sample is formed by the product of different components. One of them, the organic matter, can be modeled as a second degree polynomial [7], whereas the rest of contaminants, will produce minimums of transmittance in different wavelengths, which can be modeled using the Fermi-Gauss approach shown before. Thus, our complete model for the transmittance curve of a real water sample is:

$$\varphi(\lambda) = (a + b \cdot \lambda + c \cdot \lambda^2) \cdot \prod_{i=1}^{N_c} T(\lambda) \quad (4)$$

where N_c is the total number of contaminants to be estimated. Note that $\varphi(\lambda)$ also depends of a number of parameters

$$(a, b, c, \lambda_{min1}, T_{min1}, \sigma_{11}, \sigma_{21}, \lambda_{min2}, T_{min2}, \sigma_{12}, \sigma_{22}, \dots, \lambda_{minN_c}, T_{minN_c}, \sigma_{1N_c}, \sigma_{2N_c}),$$

(see Equation (3)), which must be optimized in such a way that the model curve to be as close as possible to the measured spectral response of the sample. Mathematically, this problem can be stated as follows:

Let φ^* be the measured spectral response of the water sample under study. We must obtain the optimal parameters

$$(a, b, c, \lambda_{min1}, T_{min1}, \sigma_{11}, \sigma_{21}, \lambda_{min2}, T_{min2}, \sigma_{12}, \sigma_{22}, \dots, \lambda_{minN_c}, T_{minN_c}, \sigma_{1N_c}, \sigma_{2N_c}) \quad (5)$$

in such a way that a given objective function is minimized:

$$f(\mathbf{x}, \lambda) = (\varphi(\lambda) - \varphi^*(\lambda))^2 \quad (6)$$

Note that, in general, we do not know the exact number of contaminants in a water sample N_c , but these can be estimated by the number of minimums in its spectral response, in the wavelengths of interest. In the next section we propose to use two heuristic techniques (a PSO and a VNS algorithms) in order to solve the problem of the optimal estimation of the model's parameters stated above.

3 Model's Parameters Estimation: PSO and VNS Approaches

3.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy [10], inspired by social behavior of bird flocking and fish schooling. A PSO system is initialized with a population of random solutions, and searches for the optimal one by updating generations. PSO has no evolution operators such as crossover and mutation as genetic algorithms do, but potential solutions instead, called *particles*, which fly through the problem search space to look for promising regions according to its own experiences and experiences of the whole group. Thus, social information is shared, and individuals profit from the discoveries and previous experiences of other particles in the search. The PSO is considered a global search algorithm.

Mathematically, given a swarm of N particles, each particle $i \in \{1, 2, \dots, N\}$ is associated with a position vector $\mathbf{x}_i = (x_1^i, x_2^i, \dots, x_K^i)$, with K the number of parameters to be optimized in the problem. Let \mathbf{p}_i be the best previous position that particle i has ever found, i.e. $\mathbf{p}_i = (p_1^i, p_2^i, \dots, p_K^i)$, and \mathbf{g} be the group's best position ever found by the algorithm, i.e $\mathbf{g} = (g_1, g_2, \dots, g_K)$. At each iteration step $k + 1$, the position vector of the i th particle is updated by adding an increment vector $\Delta \mathbf{x}_i(k + 1)$, called velocity $\mathbf{v}_i(k + 1)$, as follows:

$$v_d^i(k + 1) = v_d^i(k) + c_1 r_1(p_d^i - x_d^i(k)) + c_2 r_2(g_d - x_d^i(k)), \quad (7)$$

$$v_d^i(k + 1) = \frac{v_d^i(k + 1) \cdot V_d^{max}}{|v_d^i(k + 1)|}, \text{ if } |v_d^i(k + 1)| > v_d^{max}, \quad (8)$$

$$x_d^i(k + 1) = x_d^i(k) + v_d^i(k + 1), \quad (9)$$

where c_1 and c_2 are two positive constants, r_1 and r_2 are two random parameters which are found uniformly within the interval $[0, 1]$, and v_d^{max} is a parameter that limits the velocity of the particle in the i th coordinate direction. This iterative process will continue until a stop criterion is satisfied, and this forms the basic iterative process of a standard PSO algorithm [10].

3.2 Variable Neighborhood Search

Variable Neighborhood Search (VNS) is a recently proposed metaheuristic [11], [12] for solving combinatorial optimization problems. It is based on the use of

several neighborhood structures \mathcal{N}_k , and to proceed to a systematic change of them within a local search. In this case we have implemented an ants algorithm as local searcher.

We consider a *restricted version* of VNS which only considers two neighborhoods \mathcal{N}_1 and \mathcal{N}_2 . The VNS algorithm repeatedly performs three steps combining stochastic and deterministic strategies: In the first one, called *shaking*, a solution \mathbf{x} is randomly generated in \mathcal{N}_k . In the second one, a local search method is applied from \mathbf{x} to obtain a local optimum solution \mathbf{x}'' . In the third one, if \mathbf{x}'' is better than \mathbf{x} , then \mathbf{x} is replaced with \mathbf{x}'' and k does not change; otherwise, k is switched (from 1 to 2 or from 2 to 1 in this case of two neighborhoods). In our VNS implementation k is initially set to 1 and the method resorts to \mathcal{N}_2 when \mathcal{N}_1 (now in combination with the local search) fails to improve on the current solution. If \mathcal{N}_2 also fails to improve on the incumbent solution, the VNS sets $k = 1$ and randomly selects another trial solution in \mathcal{N}_1 , repeating the three steps again. The sequence is repeated until a *Max_iter* number of consecutive iterations is performed with no further improvement.

Consider then an initial solution \mathbf{x} , for our problem, where each x_i are the components of vector \mathbf{x} , $x_i \in [\text{lim_inf}_i, \text{lim_sup}_i]$. Our VNS algorithm defines \mathcal{N}_1 as all the feasible values for the different components which form a solution \mathbf{x} , i.e. $x_i \in [\text{lim_inf}_i, \text{lim_sup}_i]$. On the other hand, \mathcal{N}_2 is formed by a reduced set of values, in order to obtain accuracy in the search. Our shaking procedure is different in \mathcal{N}_1 and in \mathcal{N}_2 . If we are in \mathcal{N}_1 , the shaking procedure consists of a random updating of the solution in all the feasible values of its components:

$$x_i = x_i + 2 \cdot (2 \cdot \alpha_{1i} \cdot \text{rand} - \alpha_{1i}), \quad (10)$$

where *rand* stands for a random uniform value between 0 and 1. The value of α_{1i} depends on the values in which parameter x_i is defined (feasible values of the variable), and of course they may be different for different x_i . Note that if $x_i > \text{lim_sup}_i$ then $x_i = \text{lim_sup}_i$ and if $x_i < \text{lim_inf}_i$ then $x_i = \text{lim_inf}_i$.

If we are in neighborhood \mathcal{N}_2 , then we define a surrounding of the current solution as $x_i \in [\text{lim_inf}_i^{\alpha_2}, \text{lim_sup}_i^{\alpha_2}]$. In this case, the shaking procedure consists of varying the value of these limits around the best solution found so far in the search: $\text{lim_inf}_i^{\alpha_2} = x_i - 2\alpha_{2i}$ and $\text{lim_sup}_i^{\alpha_2} = x_i + 2\alpha_{2i}$, with α_{2i} small.

As we have mentioned above, the local search method in this VNS algorithm is an ants scheme similar to the one described in [14]. A number m of ants are randomly distributed over the current neighborhood. Then modifications based on the pheromone trail are applied to each ant. Following [14] the amount of pheromone only intensifies around the best objective function value obtained from the previous iteration and all ants turned towards that point to keep searching a solution. Each ant is updated at the beginning of each iteration, using the following Equation:

$$x_t^k = x_{t-1}^{\text{best}} \pm dx \quad (11)$$

where x_t^k at iteration t , x_{t-1}^{best} is the best solution obtained at iteration $t-1$ and dx is a randomly generated vector in the current neighborhood. Regarding the

pheromone (τ_t) updating, its quantity is reduced to simulate the evaporation process following this equation:

$$\tau_t = 0.1 \cdot \tau_{t-1} \quad (12)$$

And, on the other hand, it is only increased around the best objective function value obtained in the previous iteration:

$$\tau_t = \tau_{t-1} + 0.01 \cdot f_t^{best} \quad (13)$$

See [13] and [14] for further details on this algorithm.

3.3 Application of the PSO and VNS to the Estimation of Nitrates

Both the PSO and VNS algorithms shown above can be directly applied to estimate the optimal parameters of our transmittance model with very few changes: First, each particle in the PSO algorithm (solution in the VNS) are in this application

$$\mathbf{x} = (a, b, c, \lambda_{min1}, T_{min1}, \sigma_{11}, \sigma_{21}, \dots, \lambda_{minN_c}, T_{minN_c}, \sigma_{1N_c}, \sigma_{2N_c}).$$

Note that we have four parameters by contaminant considered, plus three of the polynomial representing the organic matter ($3 + 4 \cdot N_c$ parameters in total). The objective function of the problem is given by Equation (6).

4 Experiments and Results

We have tested the performance of the proposed PSO and VNS in the analysis of 10 laboratory samples, and two real spectroscopy transmittance measures obtained in two different aquifers of Madrid, Spain. In all cases an optic spectrophotometer Stellarnet based on a diffraction grating has been used. The obtained data have been treated using the software Spectrawiz 2.1. This spectrophotometer allows measures in the range of wavelength 200-800 nm., with a precision of 0.5 nm. Before performing the measures, the spectrophotometer must be calibrated using an additional pattern.

The laboratory samples consist of pure water with a given (known) concentration of nitrates, from 10 ppm to 100 ppm, in steps of 10. Since a simple contaminant conforms the sample, the transmittance curve can be modeled using 7 parameters. We have applied the PSO and VNS described in Section 3 to the estimation of these samples. Table 1 shows the values of the quadratic error (given by Equation (6)) between the model and each laboratory sample, obtained with the PSO and VNS algorithms. Both algorithms were run with a similar number of function evaluations (about 200000), in order to fairly compare them. Note that the VNS algorithm outperform to the PSO, and obtains better results in terms of the fitness function used.

We have also tested the PSO and VNS algorithms in two real samples corresponding to a waste-water land-application plant sited in Redueña, at the north

Table 1. Performance comparison between the PSO and VNS in terms of the objective function used (Equation (6)) (quadratic mean error over all wavelengths)

Sample	PSO	VNS
10 ppm	4.875	2.343
20 ppm	3.267	1.795
30 ppm	7.369	2.097
40 ppm	5.359	2.123
50 ppm	5.613	1.733
60 ppm	4.259	1.811
70 ppm	4.081	1.791
80 ppm	6.027	2.311
90 ppm	4.265	2.013
100 ppm	4.599	1.530

of Madrid, and a measure in a well in Loranca, at the south of Madrid. Figures 2 (a), (b) show the transmittance measures obtained in Redueña and Loranca, and also the modeled transmittances obtained with the VNS algorithm. Both samples have been modeled considering 3 contaminants (15 parameters). The measure error (given by Equation 6) when using the PSO is 9.23 and 6.50, whereas using the VNS is 2.91 and 1.26 for the samples of Redueña and Fuenlabrada, respectively. Note that also in these real samples the VSN approach obtains better results than the PSO.

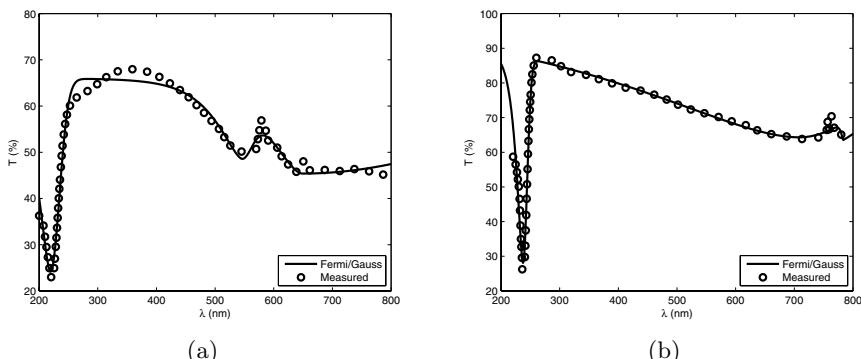
**Fig. 2.** Real transmittance curves measured in the considered aquifers; (a) Transmittance curve obtained in Redueña; (b) Transmittance curve obtained in Loranca

Figure 3 displays the nitrates component of the modeled Fermi-Gauss transmittance for the Redueña's water sample. This component indicates a nitrates concentration of 15 ppm. A similar analysis can be done for the Loranca's water sample, given a concentration of 6 ppm. The nitrates concentration in Redueña is higher than in Loranca, since Redueña's sample comes from a waste-water

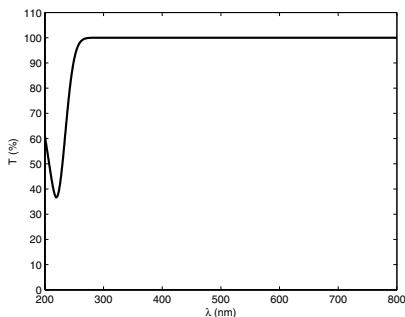


Fig. 3. Modeled curve component of nitrate for the Redueña water sample

land-application plant, whereas Loranca's sample was measured in a well. These results show that the proposed method can be applied to obtain an indicative measure of the nitrates concentration in water. The method could be applied to any other type of contaminant with a known transmittance curve model.

5 Conclusions

In this paper we have proposed a novel model for estimating the concentration of nitrates in water. Our approach considers that transmittance curves of samples containing nitrates can be modeled using a product of polynomial, Fermi and Gauss functions. We adjust the parameters of the model using heuristics algorithms (a PSO and VNS in this case), in such a way that the differences between the model curve and the real (measured) curve should be minimum. We have tested our approach in several transmittance curves samples from laboratory and real ones obtained in different aquifers of Madrid. The application of our method have discovered moderate to high concentrations of nitrate in the real samples, mainly in samples from a waste-water land-application plant. Our method can be extended to different types of contaminants if their minimum of transmittance is known.

References

1. Hu, Y., Liu, J., Li, W.: Resolution of overlapping spectra by curve-fitting. *Analytica Chimica Acta* 538, 383–389 (2005)
2. Wolfe, A., Patz, J.A.: Reactive nitrogen and human health: acute and long-term implications. *Ambio*. 31(2), 120–125 (2002)
3. Almasri, M.N., Kaluarachchi, J.J.: Modeling nitrate contamination of groundwater in agricultural watersheds. *Journal of Hydrology* 343, 211–229 (2007)
4. Holm, T., Kelly, W., Sievers, L., Webb, D.: A comparison of ultraviolet spectrophotometry with other methods for the determination of nitrate in water. *Spectroscopy* 12(9), 38–48 (1997)

5. Gross, A., Boyd, C., Seo, J.: Evaluation of the ultraviolet spectrophotometry method for the measurement of total nitrogen in water. *Journal of the World Aquaculture Society* 30(3), 388–393 (1999)
6. Johnson, K., Coletti, L.: In-situ ultraviolet spectrophotometry for high resolution and long-term monitoring of nitrate, bromide and bisulfide in the ocean. *Deep-sea Research Part I-Oceanographic research* 49(7), 1291–1305 (2002)
7. Thomas, O., Gallot, S., Mazas, N.: Ultraviolet multiwavelength absorptiometry (Uvma) for the examination of natural-waters and wastewaters I: general considerations. *Fresenius Journal of Analytical Chemistry* 338(3), 234–237 (1990)
8. Thomas, O., Gallot, S., Mazas, N.: Ultraviolet multiwavelength absorptiometry (Uvma) for the examination of natural-waters and wastewaters II: determination of nitrate. *Fresenius Journal of Analytical Chemistry* 338(3), 238–240 (1990)
9. Shrestha, R., Bárdossy, A., Rode, M.: A hybrid deterministic-fuzzy rule based model for catchment scale nitrate dynamics. *Journal of Hydrology* 342, 143–156 (2007)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of the IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
11. Hansen, P., Mladenovic, N.: An introduction to VNS. In: Metaheuristics Advances and Trends in Local Search Paradigm for Optimization, pp. 433–458. Kluwer, Dordrecht (1998)
12. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. *European Journal of Operational Research* 130, 449–467 (2001)
13. Toksari, M.D., Güner, E.: Solving the unconstrained optimization problem by a variable neighborhood search. *Journal of Mathematical Analysis and Applications* 328, 1178–1187 (2007)
14. Toksari, M.D.: Ant colony optimization for finding the global minimum. *Applied Mathematics and Computation* 176, 308–316 (2006)

Optimal Irrigation Scheduling with Evolutionary Algorithms

Michael de Paly and Andreas Zell

Center for Bioinformatics Tübingen (ZBIT),
72076 Tübingen, Germany

{michael.depaly, andreas.zell}@uni-tuebingen.de
<http://www.ra.cs.uni-tuebingen.de/>

Abstract. Efficient irrigation is becoming a necessity in order to cope with the aggravating water shortage while simultaneously securing the increasing world population's food supply. In this paper, we compare five Evolutionary Algorithms (real valued Genetic Algorithm, Particle Swarm Optimization, Differential Evolution, and two Evolution Strategy-based Algorithms) on the problem of optimal deficit irrigation. We also introduce three different constraint handling strategies that deal with the constraints which arise from the limited amount of irrigation water. We show that Differential Evolution and Particle Swarm Optimization are able to optimize irrigation schedules achieving results which are extremely close to the theoretical optimum.

Keywords: Irrigation scheduling, deficit irrigation, evolutionary computation.

1 Introduction

Water is more and more becoming a limited resource. This is especially the case in arid regions where irrigated agriculture is the largest contributor to water waste and at the same time represents the main source for food. Therefore the Food and Agriculture Organization of the United Nations (FAO) called for a revolution in water management and usage efficiency[1].

A novel approach to increase irrigation efficiency is deficit irrigation which consciously undersupplies the plants with water and accepts reduced crop yields. Since for many crops like maize the relationship between irrigation water and yield is highly nonlinear, even a significant reduction of irrigation water only results in a comparably small yield loss. However, the sensitivity of most crops to water stress changes dramatically over the different growth phases. It is therefore important to plan irrigations beforehand with the changing water needs of the crop and the amount of available irrigation water in mind in order to avoid overirrigation on one hand and catastrophic yield loss on the other hand.

Many current deficit irrigation tools like CROPWAT[2] are limited to a simulated sensor controlled irrigation scheduling based on water balance models. Since such strategies do not pay attention to the interdependency of irrigation

events they fail to achieve the best possible yield from a given amount of water. To tackle this problem Bras et al. proposed a global approach to irrigation scheduling based on dynamic programming (DP)[3]. The use of DP imposed severe limitations on the problem. Irrigations could only be scheduled on a very limited number of predetermined irrigation days, the underlying water balance model had to be coarsely discretized, and an extremely simplified yield prediction model had to be used, which fails to take the relationship between growth period and water stress sensitivity into account. This oversimplification has been criticized by Rao, Sarma, and Chander who proposed a division into separate optimization runs for each growth phase to allow the usage of more sophisticated yield prediction models [4,5].

A further improvement of the DP approach described by Sunantara et al. removed the separation of the optimization stages and allowed for daily irrigation decisions [6]. However, the necessary coarse discretization of the underlying water balance and yield prediction models remained. This seriously limits the predictive capabilities of the models and subsequently affects the resulting irrigation schedules. Using Evolutionary Algorithms for irrigation scheduling as proposed by Schütze and Schmitz circumvents these limitations and allows the use of almost arbitrary complex models [7,8].

For this paper we therefore systematically evaluated the performance of Evolutionary Algorithms on deficit irrigation scheduling. Five Evolutionary Algorithms (real valued Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, standard and covariance matrix adaption Evolutionary Strategy) were combined with three different constraint handling strategies and applied to a deficit irrigation scheduling problem based on a real valued water balance and yield prediction model. The two best performing algorithms and the best performing constraint handling strategy were selected and investigated in more detail.

2 Methods

2.1 Water Balance Model

To simulate the water balance in the field we used the water balance model described in [9]. The basic idea of this model is the consideration of the soil as a storage system which is characterized by the relative soil moisture Θ_i and depth of the root zone D_i for each day i . The upper limit of the storage capacity is defined by the field capacity fc . Excess water that would result in $\Theta_i > fc$ can not be held and is lost through surface runoff or percolation. The lower limit is defined by the permanent wilting point pwp , which defines the minimal necessary soil moisture to prevent unrecoverable wilting of the crop resulting in a complete yield loss. Water transport processes in the field are modeled as sources and sinks as given in:

$$\Theta_i = \min(fc, \max(pwp, \frac{\Theta_{i-1}D_{i-1} + I_i + P_i - AET_i + \Theta_0(D_i - D_{i-1})}{D_i})) \quad (1)$$

where P_i is the precipitation, I_i is the irrigation water, and AET_i is the actual evapotranspiration, which is the sum of evaporation and transpiration by the plants. AET_i , which depends on the type of the crop and the potential evapotranspiration PET_i , is calculated according to:

$$AET_i = \begin{cases} PET_i & \Theta_i D_i \geq (1 - p_i(PET_i))(fc - pwp)D_i \\ \frac{\Theta_i D_i PET_i}{(1 - p_i(PET_i))(fc - pwp)D_i} & \Theta_i D_i < (1 - p_i(PET_i))(fc - pwp)D_i \end{cases} \quad (2)$$

where p_i is the crop dependent soil water depletion factor, which is a function that describes the relationship between water uptake of the plants and PET . Iterative solving of (1) and (2) allows the simulation of the development of AET for an entire growth period, which is the base for the subsequent yield prediction model.

2.2 Yield Prediction

To obtain a yield estimate we used a multiplicative approach based on the yield prediction model published by the FAO in [10], which calculates the actual yield Y_s for a given crop according to:

$$Y_s = 1 - K_y(s) \frac{AET_s}{PET_s} \quad (3)$$

where AET_s and PET_s are the integrated actual and potential evapotranspiration for the growth period s . K_y is the yield response factor, which describes the sensitivity of a given crop to water stress. Since K_y varies heavily over time depending on the growth stage, we calculated independent relative yields Y_s according to (3) for four distinctive growth stages $s = 1 \dots 4$. The overall yield Y can then be derived by multiplying the product of the relative yields for all stages with Y_m , which is the maximum obtainable yield under the assumption of optimal growing conditions:

$$Y = Y_m \prod_s Y_s \quad (4)$$

2.3 Fitness Function and Constraints

The combination of water balance and yield prediction model, given above, establishes a relationship between an irrigation schedule I and the resulting crop yield $Y(I)$, where I is a vector containing the irrigation amounts I_i for all days of the whole growth period. To evaluate the fitness of an irrigation schedule we calculated the yield loss Y_l according to:

$$f(I) = Y_l(I) = Y_m - Y(I) \quad (5)$$

A hallmark of deficit irrigation is the limited amount of water which restricts the water use of a schedule to a maximum amount Q_{max} . Therefore the cumulated irrigation water of a schedule I is subject to the inequality constraint:

$$\sum_{\forall I_i \in I} I_i \leq Q_{max} \quad (6)$$

For obvious reasons all irrigation events in an irrigation schedule also have a lower boundary of 0:

$$I_i \geq 0 \quad \text{for } \forall I_i \in I . \quad (7)$$

2.4 Constraint Handling

Since the Evolutionary Algorithms considered in this paper are geared towards unconstrained optimization problems, additional measures to handle the maximum water constraint have to be taken. We considered three different strategies. The first one (Penalty) allowed any amount of water in a given schedule but penalized the fitness of schedules which violate the maximum water constraint depending on the amount of water above the allowed upper limit:

$$f(I) = Y_l(I) + \begin{cases} 0 & \sum_i I_i \leq Q_{max} \\ 0.5 - Q_{max} + \sum_i I_i & \sum_i I_i > Q_{max} \end{cases} . \quad (8)$$

The second (Repair) and third (Lamarck) strategy exploit the semilinear relationship between the amount of irrigation water and yield to repair invalid schedules. Under the assumption of optimal scheduling additional irrigation water always leads to improved yield as long as the water requirements of the crop are not totally fulfilled. It can therefore be assumed that optimal deficit irrigation schedules always use the maximum available irrigation water. Based on this assumption invalid schedules can be repaired by forcing the amount of used irrigation water to the maximum available irrigation water through normalization with Q_{max} :

$$I'_i = I_i * \frac{Q_{max}}{\sum_i I_i} . \quad (9)$$

The Repair and Lamarck strategy differ in the way repaired individuals are handled. Repair uses repaired individuals only for fitness calculations and keeps the population unchanged while Lamarck additionally uses a lamarckian approach which replaces invalid individuals in the population with their repaired counterparts.

2.5 Algorithms

Based on the Evolutionary Algorithm framework Eva2, which is the successor of JavaEva [11]¹, we tested the following Evolutionary Algorithms on the yield loss minimization problem:

1. Real valued Genetic Algorithm (realGA) with global mutation ($p_m = 0.1$) and uniform crossover ($p_c = 0.9$) using a population size of $n = 50$
2. Particle Swarm Optimization (PSO) in constriction mode with $\phi_1 = \phi_2 = 2.1$ and a population size of $n = 30$ in a grid topology.

¹ Eva2 and JavaEva have been developed in our research group and are freely available under a LGPL license see <http://www.ra.cs.uni-tuebingen.de/software/Eva2/index.html>

3. Differential Evolution (DE) with $F = 0.8$, $C_r = 0.6$ and a population size of $n = 30$
4. Evolution Strategy (ES) (5,20) and (5+20) with global mutation ($p_m = 0.8$) and one-point crossover ($p_c = 0.2$)
5. Covariance Matrix Adaption Evolution Strategy (cmaES) (5,20) and (5+20) without crossover ($p_c = 0$ and $p_m = 1$)

All algorithms were tested in combination with the three constraint handling strategies described above. To obtain statistically significant results each experiment was repeated 20 times. The number of fitness evaluations had to be limited to 5000 for each experiment in order to keep the overall runtime feasible.

2.6 Scenario

To compare the various Evolutionary Algorithms we assumed a scenario based on a field with a soil of silty loam resulting in a field capacity of $fc = 0.4$ and a permanent wilting point of $pwp = 0.05$. The crop used was maize with the soil water depletion factors p , yield response factors K_y , and root zone depth d_i given in [10]. ETP data has been taken from field experiments by J.C. Mailhol from CEMAGREF in Montpellier. The crop water requirement for optimal growth conditions is approximately 54.3 cm for this scenario. To simulate deficit irrigation we limited the maximum available irrigation water to 85% of this value respectively 46.15 cm. To ease comparison, Y_m has been set to 1. The minimal achievable yield loss in this case is 7.5%.

3 Comparison Results

Figure 1 and Table 1 show the the results for each combination of optimization algorithm and constraint handling strategy. In combination with the Penalty strategy all algorithms performed considerably worse than in combination with the Repair or the Lamarck strategy. The only optimization algorithm which achieved acceptable results with the Penalty strategy is PSO. (5,20)-ES and (5,20)-cmaES even failed to find any valid irrigation schedules at all, i.e. schedules that fulfill the maximum water constraint. The comparison of the Repair and the Lamarck strategy shows a mixed picture. In combination with PSO and DE Lamarck achieved slightly better results than Repair while performing worse for all ES-based algorithms. Independently from the constraint handling strategy PSO and DE always performed better than realGA and the ES-based algorithms.

One reason for the low performance of the ES based algorithms may have been the low amount of fitness evaluations. We therefore did additional experiments on the (5,20)-ES and (5+20)-ES combined with the Repair and the Lamarck strategy using 50000 function evaluations. As Fig. 2 shows, the higher number of function evaluations did not result in a significant reduction of Y_l for the Repair strategy. For Lamarck the increased number of function evaluations led to a slight improvement of the results achieving but not surpassing the results of the Repair strategy in the case of (5,20)-ES. (5+20)-ES did not finally converge

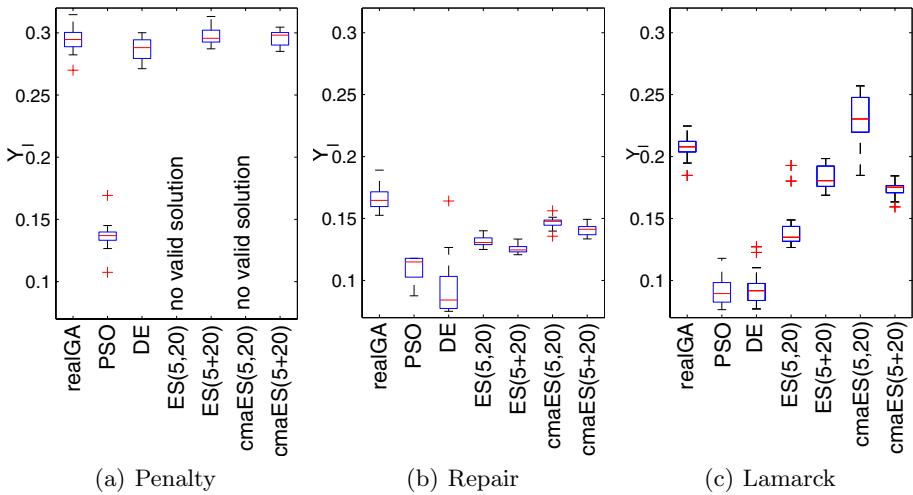


Fig. 1. comparison of achieved yield loss for the constraint handling strategies

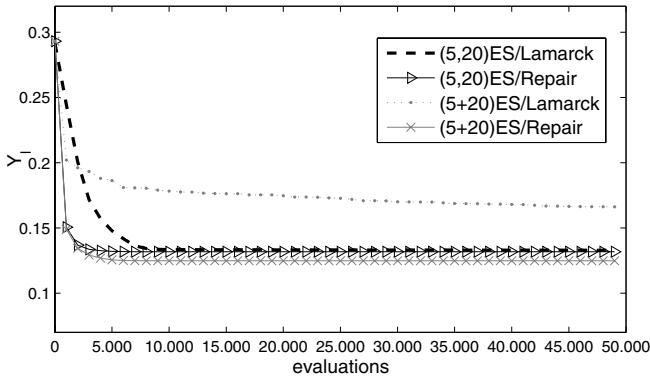


Fig. 2. Development of Y_l for (5,20)-ES and (5+20)-ES

with Lamarck even with a 10 folded increase of function evaluations. We suspect that the unsatisfactory performance of the surveyed ES-based algorithms can be attributed to the extremely multi modal target function which causes premature convergence of the ES based algorithms due to the high selection pressure of ES which makes it hard to leave local minima.

4 Fine Tuning DE and PSO

4.1 Optimized Settings for the Optimization Algorithms

As the previous section showed, PSO and DE achieved the best results for this problem. For that reason we conducted further tests fine tuning the settings of

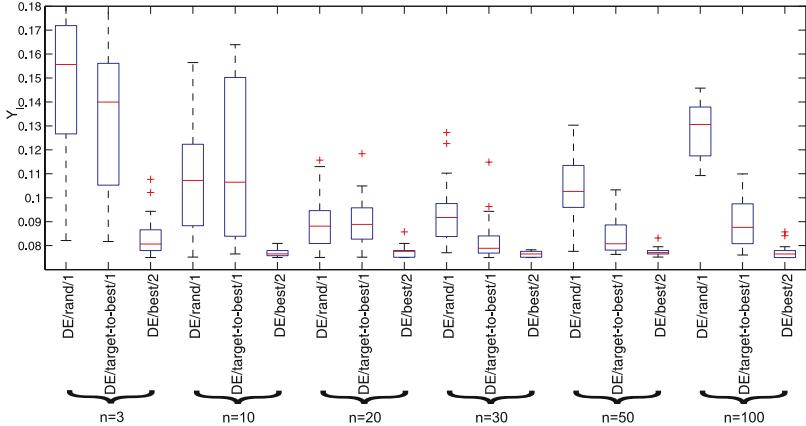


Fig. 3. Benchmark of DE type and population size n with $F = 0.8$ and $C_r = 0.6$

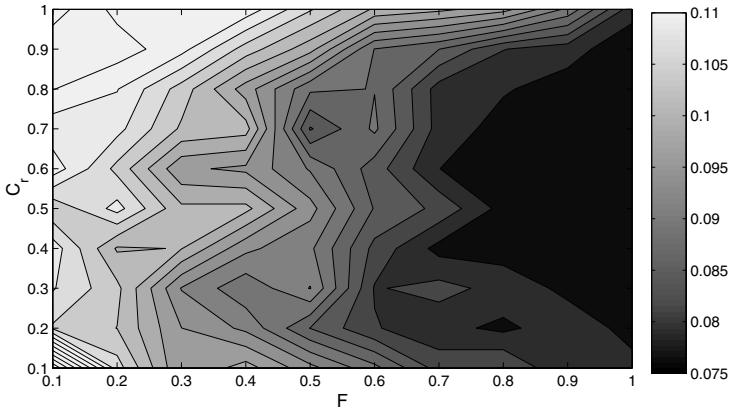
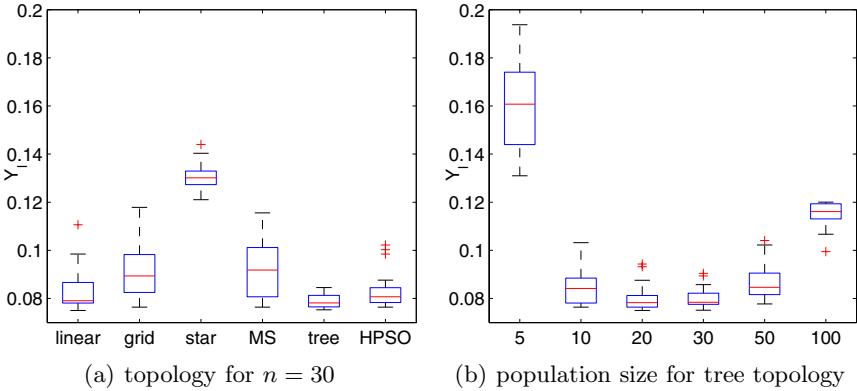


Fig. 4. Impact of F and C_r on the performance of DE/best/2 with $n = 30$. Lower yield values (darker regions) are better.

both algorithms. We used the same scenario as in the experiments described above. As before, the number of fitness evaluations was limited to 5000 and each experiment was repeated 20 times for each setup. Because of its superior performance in combination with PSO and DE constraint handling was done with Lamarck.

For DE we considered the types DE/rand/1, DE/target-to-best/1, and DE/best/2 which are described in [12]. Beside the type we also varied the population size $n \in \{3, 10, 20, 30, 50, 100\}$. For the most promising combination of DE type and population size we did a grid search for $F, C_r \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

To determine optimal settings for PSO we tested the impact of several topologies on the performance. These were ring topology with range = 2, grid

**Fig. 5.** Benchmark of PSO parameters**Table 1.** Comparison of optimization results as normalized yield loss Y_l in %. Entries marked with *) did not converge.

Constr. handling	Penalty			Repair			Lamarck		
	Algorithm	Best	Avg.	Std. .	Best	Avg.	Std.	Best	Avg.
realGA	26.9	29.4	1.0	15.2	16.6	0.9	18.4	20.7	0.8
PSO	10.7	13.6	1.1	8.7	10.9	0.9	7.6	9.2	1.2
DE	27.1	28.7	0.8	7.5	9.4	2.2	7.7	9.3	1.3
(5,20)-ES	*)			12.5	13.1	0.4	12.6	14.0	1.7
(5+20)-ES	28.7	29.7	0.7	12.0	12.5	0.2	16.8	18.3	0.9
(5,20)-cmaES	*)			13.5	14.6	0.4	18.4	22.8	2.1
(5+20)-cmaES	28.5	29.5	0.6	13.3	14.0	0.3	15.9	17.3	0.6
PSO optimized	tree topology $n = 20$						7.5	8.0	0.6
DE optimized	DE/best/2 $n = 30 F = 0.9 C_r = 0.6$						7.5	7.6	0.1

topology with range = 2, star topology, self organized multi-swarm topology with sub swarm size = 0.2, tree topology with branching factor = 3, and a hierarchical PSO (HPSO) with range = 2. For the most promising topology the influence of the population size was tested by varying n from 5 to 100.

4.2 Results of Optimized DE and PSO

As Fig. 3 shows, DE/best/2 outperformed DE/rand/1 and DE/target-to-best/1 by a considerable margin for all examined population sizes n . While the population size has a large impact on the performance of DE/rand/1 and DE/target-to-best/1 with an optimum for values of n between 20 and 30, the influence of n on the results of DE/best/2 is extremely small for a wide range of n . We therefore selected DE/best/2 with $n = 30$ for further analysis. Figure 4 shows the relationship between F, C_r and the achieved Y_l which has been averaged over 20 optimization runs. While F has a strong impact on the performance, the

influence of C_r is much less pronounced. This is especially true for F close to 1 where a wide range of values of C_r produced similar results.

The comparison of the various PSO topologies (Fig. 5(a)) indicates a strong relationship between topology and optimization performance, with the star topology delivering the worst and the tree topology along with the linear topology delivering the best results. According to Fig. 5(b), population sizes of 20 and 30 turn out to be the most effective for a tree topology. Small population sizes especially below 10 seem to lead to premature convergence resulting in a high variance of Y_l depending on the initial population. Population sizes above 30 result in a downgraded performance which can most probably be attributed to the lower amount of generations caused by the larger amount of function evaluations per generation.

5 Conclusion

In this paper we have presented a systematic survey of Evolutionary Algorithms on a deficit irrigation problem. We also introduced three constraint handling strategies which ensure the adherence of the optimized irrigation schedules to the limit of the available irrigation water. These were a penalty strategy as well as a repair strategy with and without lamarckian evolution. Furthermore we fine tuned the settings of the two best performing algorithms namely PSO and DE in combination with the Lamarck strategy.

The experiments in the paper showed that both algorithms are able to solve the deficit irrigation problem with excellent results. With a mean of 7.6% the yield losses achieved by the fine tuned DE are extremely close to the theoretical optimum of 7.5%. The low variance of the optimized yields also indicate a high reliability of the proposed methods. Unlike DE and PSO the ES-based algorithms and realGA performed not as well and did not achieve comparable results even when allowed to use a much larger number of fitness evaluations than DE and PSO.

The Evolutionary Algorithms examined in this paper do not make any assumptions about the inner workings of the underlying irrigation and yield prediction models. Therefore the methodology used in this paper should be applicable to a wide range of models and not be bound to water balance models. Further extensions of this approach will make use of more complex models which also simulate the impact of fertilizers on crop growth and allow the evaluation of the ecological footprint of an irrigation strategy.

References

1. Diouf, J.: Agriculture, food security and water: Towards a blue revolution, *OECD Observer* (2003)
2. Smith, M.: Cropwat: A computer program for irrigation planing and management, Tech. report, FAO Land and Water Development Division, Rome (1992)
3. Bras, R.L., Cordova, J.R.: Intraseasonal Water Allocation in Deficit Irrigation. *Water Resources Research* 17(4), 866–874 (1981)

4. Rao, N.H., Sarma, P.B.S., Chander, S.: Irrigation Scheduling under a Limited Water Supply. *Agricultural Water Management* 15, 168–175 (1988)
5. Rao, N.H., Sarma, P.B.S., Chander, S.: Optimal Multicrop Allocation of Seasonal and Intraseasonal Irrigation Water. *Water Resources Research* 26(4), 551–559 (1990)
6. Sunantara, J.D., Ramirez, J.A.: Optimal Stochastic Multicrop Seasonal and Intraseasonal Irrigation Control. *Journal of Water Ressource Planing and Management* 123(1), 39–48 (1997)
7. Schütze, N., Wöhling, T., de Paly, M., Schmitz, G.: Global optimization of deficit irrigation systems using evolutionary algorithms. In: *Proceedings of the XVI International Conference on Computational Methods in Water Resources*, Copenhagen, Denmark (2006)
8. Schmitz, G., Wöhling, T., de Paly, M., Schütze, N.: Gain-p: A new strategy to increase furrow irrigation efficiency. *Arabian Journal for Science and Engineering* 32(1C), 103–114 (2007)
9. Rao, N.H.: Field test of a simple soil-water balance model for irrigated areas. *Journal of Hydrology* 91, 179–186 (1987)
10. Doorenbos, J., Kassam, A.H.: Yield response to water. *FAO Irrigation and Drainage Paper* 33, FAO Rome (1979)
11. Streichert, F., Ulmer, H.: JavaEvA - A Java Framework for Evolutionary Algorithms. Center for Bioinformatics Tübingen, University of Tübingen, Technical Report WSI-2005-06 (2005), <http://w210.ub.uni-tuebingen.de/dbt/volltexte/2005/1702/>
12. Price, K., Storn, R.M., Lampinen, J.A.: *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, New York (2005)

Adaptive Land-Use Management in Dynamic Ecological System

Nanlin Jin¹, Daniel S. Chapman², and Klaus Hubacek¹

¹ University of Leeds, Leeds, UK, LS2 9JT

² The Centre for Ecology & Hydrology, Midlothian, UK, EH26 0QB
 {n.jin,leckh}@leeds.ac.uk, dcha@ceh.ac.uk

Abstract. UK uplands are significantly important in the economy and the environment. There is also a debate on how the banning of managed burning will affect the landscape of uplands. One difficulty in answering such a question comes from the fact that land-use management continuously adapts to dynamic biological environments, which in turn have many impacts on land-use decisions. This work demonstrates how evolutionary algorithms generate land-use strategies in dynamic biological environments over time. It also illustrates the influences on sheep grazing from banning managed burning in a study site.

Keywords: Genetic algorithms, dynamic optimization.

1 Introduction

In the Peak District National Park, one of the UK's uplands, sheep grazing and grouse hunting have been two major land-uses for more than one thousand years. These two land-uses are partly due to the vegetation cover in this area. The majority area is covered by grass and heather [7]. Grass is a food source for sheep, while heather is an ideal habitat for grouse. Land managers' income is mainly from selling lamb and charging license fees of grouse hunting [24].

In order to maintain sustainability, land managers must consider both land management strategies, and their impacts on environments. It is widely appreciated that management shapes the dynamics of ecological systems, while ecological conditions influence management decisions. The difficulty of making sustainable land-use strategies lies in the uncertainties and dynamics between land management and biological responses. Studying management or biological dynamics alone is not sufficient to understand such dynamics [23]. We try to understand such dynamics by integrating management and biological models. We will achieve this goal through answering two questions for the banning burning scenario: (1) what would be ideal grazing densities; (2) what would be the vegetation changes as the result of no managed burning. To answer these two questions will also help us understand how land-use decisions and biological responses inter-depend on each other. Fig. 1 illustrates the simplified relationship between the management model and the biological model.

To start with, we will briefly describe the biology model which focuses on (1) land-use impacts on vegetation and (2) the relationship between sheep density and grouse population to vegetation. The biology model studies four types of land covers: heather,

bracken, grass, bare peat and rock. Here we simplify land cover into two types: grass which is for sheep grazing, and heather which is the habitat for grouse. We develop a biological model which simulates the vegetation dynamics caused by geology, climate and land-use. We assume that geology and climate are static during the study period, while land-use management is evolving over time. Land-use and vegetation cover determine the changes of vegetation in future. Vegetation cover directly constrains the minimum number of sheep stocking density, and partly determines the grouse population. Competition between heather and grass is mediated by grazing pressure, heather age (determined by the burning rotation) and environmental gradients. Land management is a major driver of ecological change, and decisions about how people use the land not only influence ecological processes, but are made with reference to the ecological state of the system. The biology model is established on the base of field work, statistical data and literature in sheep density models and grouse population models [23]. The biological model can be viewed as an expert system.

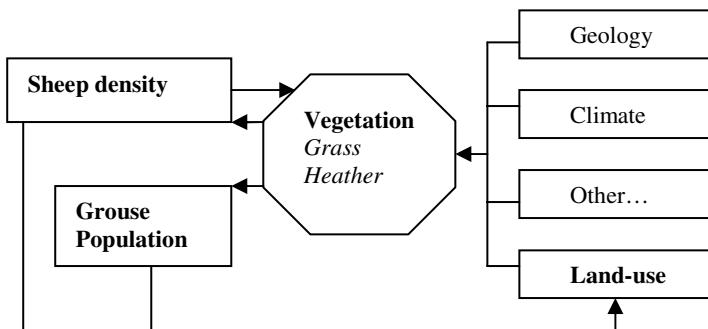


Fig. 1. Integration of the management model and the biological model

The management model emphasizes how to make land-use decisions subject to economic considerations, policy changes, vegetation, and current sheep density and grouse population. In our participatory studies, such factors are major considerations in any land-use decision for farming in this area. As explained before in the biology model, such factors are dynamic and related to land-use. Such inter-dependence of dynamics makes the system even more complex. External factors, such as farming policy change, economic recession or climate change add further uncertainties to the system. In this paper, we assume they are static during the study period.

In most literature, these two models work independently. When biological models need information from the management part, they apply static findings. Similarly when management models request information from the biological part, such models take known results on to it. Little literature actually integrates dynamic biological model and dynamic management model together or runs them in a repeating loop, simulating the cycle of land-use decision making and natural environment changes.

Models of reciprocal human-ecological interactions can produce complex and unpredictable dynamics [20]. Therefore, in order to accurately predict biodiversity responses to alterations in external policy environments or climate change, it is necessary

to simulate both the ecological and human parts of the eco-system. Integrating models to link management and ecology simulate the reality better and thus produces better predictions about how an external factor will affect the ecosystems.

2 GA-Based Management Model and Biological Models

In this section, we introduce management model and biological models. For this study site, grazing density and maintaining heather are key land-uses. We focus on modeling of grazing and heather maintenance.

Grazing density determines the number of sheep grazed per ha. From farming experiences [7], a very high grazing density leads to grass disappearance and thus erosion. Lacking a sufficient food source, the lamb production will reduce. A very low grazing density will also result in less lamb output, and heather overspread, which reduces the grass cover areas, and again brings problems in earnings.

Managed burning is carried out to encourage the regeneration of young heather shoots and to develop a mosaic of differently-aged stands that provides food and suitable habitat for grouse [22]. Low-temperature burning over appropriate rotations causes dwarf shrubs to increase, since a physiologically young and vigorous age structure is maintained [21]. The possible effects of this on biodiversity and water quality mean there is controversy over whether burning is now too intensive [7], particularly in respect of carbon emissions.

2.1 Biological Model

The biological models capture the dynamic properties and uncertainties of the vegetation model, sheep grazing and grouse population over discrete annual time steps.

At the start of each year, the management strategy employed in each management zone is decided upon, based on its vegetation composition (Fig. 2). If the strategy includes burning, this is then carried out. The grazing pressures in each cell of the zones are then determined and used to predict changes in vegetation cover. Because the flock of sheep released into each zone move freely throughout the zone, grazing pressure is not distributed evenly. The biological model produces monthly biomass, heather ages, sheep distribution and vegetation cover [23].

2.2 GA-Based Management Model

A genetic algorithm (GA) has been used for environmental management, land-use planning [13] and species distribution [15]. These works focus on static optimization. This work emphasizes an important feature in optimization: dynamic optimization.

In dynamic optimization, the optima change over time. It is necessary to track the optima continuously. Evolutionary algorithms can address optimization problems with dynamics [12]. Evolutionary algorithms simulate natural evolution [9]. Natural evolution continuously adapts to dynamics. The characteristic for evolutionary algorithms handling dynamic environments is that its fitness function $F_t(x)$ is time-varying, dependent on time t , so evolutionary algorithms search the changing optima continuously. The fitness of a land-use strategy has a connection with resulting vegetation in previous years. We will define a fitness function that reflects such a relationship.

Table 1. Variables and their ranges in land-use management

Variable	Range
Summer sheep density (ha^{-1})	{0, 0.5, 1, 2, 3}
Winter stocking rate	{0, 0.5, 1, 2, 3}
Frequency of managed burning (years)	{0, 1, 2, 5}
% burnt each time	{5, 10, 15}
Mean stocking density (ha^{-1})	[0, 3]
Heather growth phase diversity	[0, 2]
Labour days $\text{ha}^{-1} \text{ year}^{-1}$ for burning	[0, 500]

Table 1 lists the land-use variables. These variables are selected as we have found that they are the main land-use practices in the study site [7]. The ranges of these variables are derived from our scoping studies [7], mainly from interviewing local land users and from observations. For example, the variable "summer sheep density" has a value from one of the five categorical numbers {0, 0.5, 1, 2, 3}. The variable "heather growth phase diversity" has a number ranging between 0 and 2. The heather growth phase diversity describes the percentage of heather in a plot, which is calculated in the biological model [23].

Strategy Representation

Land-use strategies applied comprise a summer and winter sheep density and a burning regime. A strategy can be represented by a string. Table 1 lists important variables in land-use management concluded from participatory interviews [7]. These variables can be divided into two groups: (1) grazing; and (2) burning. Sheep grazing and grouse hunting are two major land-uses in this site. They are also major sources of land managers' income. The ranges of these variables come from land managers' answers to our questionnaires. Table 1 lists the variables and their ranges. For example, a GA string

0	0	1	5	0	0.842	167
---	---	---	---	---	-------	-----

represents that no grazing on this area as there is 0 grazing density in both summer and winter. This area is burnt every year with 5% burnt. The heather growth phase diversity is 0.842. For the managed burn, 167 labor days per ha per year are needed.

GA Set-Up

The parameters of the GA set-up are in Table 2. The set of candidate solutions is called "population" in terms of GA. We select GA strings for reproduction by using 3-member tournament selection. Three GA strings are randomly selected and the string that has the highest fitness is selected for reproduction. Two selected GA strings produce two new GA strings by one-point crossover. Two selected GA strings are divided at the crossover point. Then the first proportion of one string is swapped with the first proportion of another string. The newly generated two strings will be members of the population of the next generation after mutation. One-point mutations operates as one

point of a GA string is altered by another value from the variable range as in Table 1. A range of values of parameters have been tested to reduce their bias on experimental results. Our system is relatively stable with these values in Table 2.

Table 2. GA-set-up

Parameter	Value
Population Size	1000
Number of Generation	40
Selection Method	3-member tournament
Crossover Method	One-point Crossover
Crossover Rate	1
Mutation Method	One-point Mutation
Mutation Rate	0.3
Number of runs	50

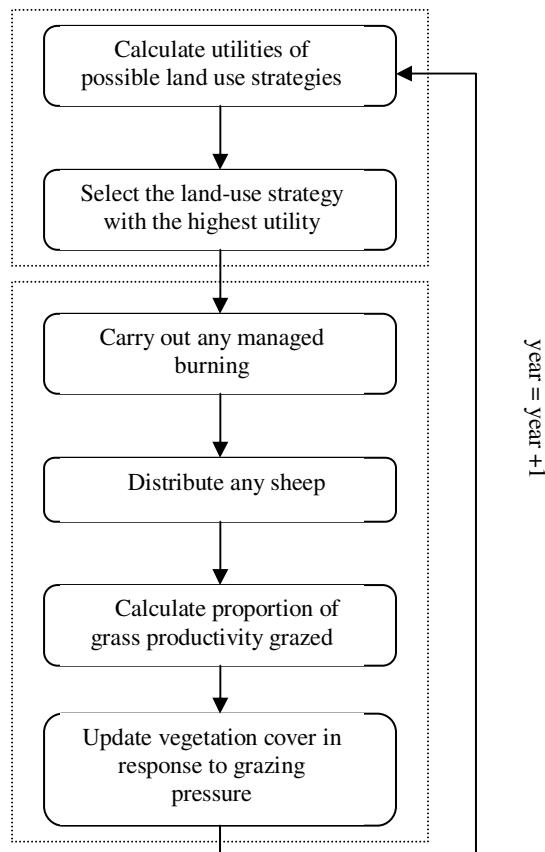


Fig. 2. Flow chart of GA-based land-use management model and the biological models

GA Process for Management model

We assume that land managers try to maximize the income for inter-temporal time T. In addition, we assume that land-use decisions must adapt to the economic and biologic conditions.

The evolutionary system simulates a land manager's decision making process. The system starts with a set (population) of mixed strategies. The objective is to find x in year t that maximizes the inter-temporal time period T. Data on T were not available and so a seemingly reasonable default value of $T = 5$ is used, with values $1 \leq T \leq 50$

explored. So the fitness function is $\sum_{t=c}^{c+T} F_t(x)$. The land manager chooses the highest

rewarding one for his inter-temporal time (for example, a 5-year inter-temporal optimi-

zation is to max $\sum_{t=c}^{c+5} F_t(x)$ and applies it to his land at year t. As mentioned before,

a strategy x is a generational replacement strategy. A year after (year $t + 1$), the land manager compares the actual income of using this strategy at year t with the possible incomes of other strategies according to his expectations of the income from year ($t+1$) to year ($t+6$). The land manager chooses the highest rewarding one among them as his decision of land-use for year ($t + 1$). Note that the fitness function is time-varying, so the same strategy may have a different fitness value in year t from its fitness in year $t + 1$.

2.3 Integrated Model

This overview tells us that vegetation and land-use dynamics are causes and results of each other. The difficulty of such a complicated system lies in there being no hard rule or formulation to predict the results. Simulation is a possible way to forecast future landscape and land-use.

Logically, the management model and biological model are integrated by their functions and roles in the ecosystem, as illustrated in Fig. 1. Physically, the two models are integrated by their inputs and outputs. The management model takes the biological model's outputs about vegetation cover and grouse population as its inputs. On the other hand, the biological model takes the management model's outputs on land-use management as its inputs.

Moreover, in the management model, the fitness function of a land-use strategy depends on both the vegetation and economic consideration. It reflects the impact of biological conditions on land managers' decision making. The utility of strategy i in management zone m , $u_{i,m}$ is then defined as,

$$u_{i,m} = 4.623\hat{S}_i - 2.404\hat{S}_i^2 - 2.225\hat{S}_i G_m \delta_{G_m \geq 0.5} - 0.472h_i + 1.297h_i D_m \delta_{D_m \geq 0.5} - 0.00135C_i \quad (1)$$

where \hat{S} is the mean (sheep) stocking density (average number of sheep ha^{-1} through the year). The range of \hat{S} is in Table 1. The summer sheep density and winter sheep density in Table 1 are not used in the fitness function, because the mean stocking density can be calculated from these two but land users prefer to use them rather than the mean stocking density. The variable h is an index of heather growth phase diversity. The variable C is the number of labour days needed for burning each year which

are retrieved from the interview data. Variables G_m and D_m are the proportions of m 's area dominated by grass and heather respectively. The variables δ_x are set to 1 if condition x is true and 0 otherwise. Model parameters were fitted to the questionnaire data using GAUSS software and a random selection of six alternative management choices for each individual observation.

Given a land-use strategy, a GP string, for example, we can calculate or directly obtain the values of variable of \hat{S} , C , h , G_m and D_m . Then, using the fitness function, we can calculate the fitness value $u_{i,m}$ of this land-use strategy.

3 Experimental Results

We have validated our integrated model by running it on a 10-year historical data on sample plots. Generally speaking, the results have revealed a land-use distribution, similar to observations found in the field. We have executed 50 GA runs which generated similar results. We are reporting the average values of these 50 runs.

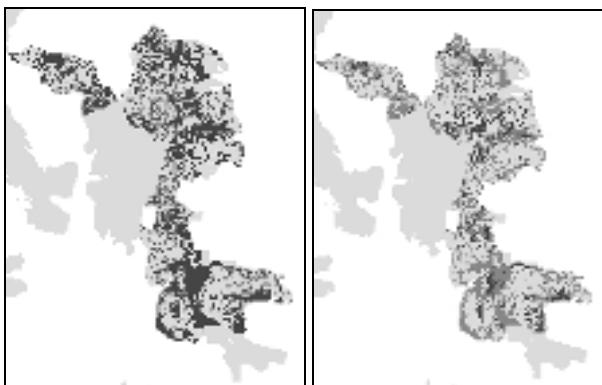


Fig. 3. Mean sheep distribution in moorland before (left) and after (right) managed burning is banned. The darker the area is the higher grazing density applies.

First of all, experimental results indicate that relatively low grazing densities are preferred. There are 71 management zones in total on this site. We start running the integrated model from applying summer grazing densities as high as 2 or 3 sheep per ha over 21 management zones, but result in none of the 71 zones applies grazing densities higher than 1. Winter grazing densities remain as low as between 0 to 1. However, more zones graze. At the beginning, there are 8 management zones have no grazing at all, but after running the model, there are only two zones which have no sheep distributed. Fig.3 illustrates that the mean grazing densities are reduced after burning is banned. We have found that the total number of sheep reduces noticeably after burning is banned. Hence there will be lower income from lamb production.

Secondly, Fig.4 illustrates that heather cover spread to some areas previously covered by grass. After burning is banned, most of heather will become over mature. Over mature heather cannot provide good food source for grouse, thus the number of grouse reduces. If there is not enough grouse to shoot, land managers can not get income from grouse hunting. However, no burning will save land managers' costs on labour.

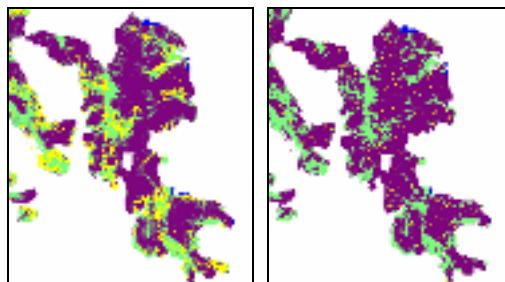


Fig. 4. Heather cover before (left) and after (right) ban burning

From the experimental results, it is obvious that the banning of managed burning will noticeably change the landscape and land-use, while land managers' income from grazing and grouse shooting shrinks.

4 Conclusion

We have attempted to understand the inter-dependent relationships between land-use management and biological environment. One of the difficulties in studying an ecosystem is rooted in the dynamics of the human system and the environment system. This paper focuses on the key dynamics in the Peak District, assuming that other dynamics remain static during the study period. Other socio-economic factors, such as subsidies and policy, and natural environment factors such as climate change, also contribute to dynamics and uncertainty for land management and ecosystem.

Questions about how a land-use change affects biological conditions have been raised. To answer these questions, we have established a model to simulate the intervention of a management model and a biological model, and applied an evolutionary algorithm to generate land-use adapting to biological changes.

This integrated model jointly simulates the dynamics of upland vegetation and its grazing and burning management. The model shows how the dynamics of management decisions influence the responses of ecosystems to an external factor, the banning on burning.

According to the experimental results, we conclude that a ban on burning will cause heather to over-mature, which is detrimental for both grouse population and sheep production. Moreover, the established model will allow us to test other scenarios and a mixture of more than one external change.

Acknowledgement

The paper has been funded through the Rural Economy and Land Use (RELU) programme, co-sponsored by Defra and SEERAD (project RES-224-25-0088). The authors would like to thank everyone involving in this project.

References

1. CAP, Single payment scheme - overview. Technical report, Department for Environment, Food and Rural Affairs, UK (November 2, 2004)
2. Bautista, R.M.: Intertemporal optimization in an agrarian model with z-activities. *Metroeconomica* 26(1-3) 200, 215 (1974)
3. Boatman, N., Ingram, J.D.J.: Agricultural change and environment observatory programme. Obs 2004. The environmental implications of the 2003 cap reforms in England. Technical report, Central Science Laboratory, York and Countryside and Community Research Unit, University of Gloucestershire (2006)
4. Chirichello, G.: Intertemporal macroeconomic models, money and rational choices. Macmillan, Basingstoke (2000)
5. Hayakama, H., Ishizawa, S.: The fundamentals of intertemporal optimization in the continuous time modelling of consumer behaviour. *Japanese Economic Review* 48(1), 101–112(12) (1997)
6. Haygarth, P.M., Chapman, P.J., Jarvis, S.C., Smith, R.V.: Phosphorus budgets for two contrasting grassland farming systems in the uk. *Soil Use and Management* 14(4), 160–167 (1998)
7. Holden, J., Shotbolt, L., Bonn, A., Burt, T.P., Chapman, P.J., Dougill, A.J., Fraser, E.D.G., Hubacek, K., Irvine, B., Kirkby, M.J., Reed, M.S., Prell, C., Stagl, S., Stringer, L.C., Turner, A., Worrall, F.: Environmental change in moorland landscapes. *Earth Science Reviews* 82, 75–100 (2007)
8. Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press (1975)
9. Holland, J.H.: *Adaptation in natural and artificial systems. An introductory analysis with applications to biology, control, and artificial intelligence*, 2nd edn. MIT Press, Cambridge (1992)
10. Hudson, P.J., Dobson, A.P., Newborn, D.: Population cycles and parasitism. *Science* 286, 2425 (1999)
11. Jin, N., Termansen, M., Hubacek, K., Holden, J., Kirkby, M.: Adaptive farming strategies for dynamic economic environments. In: *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*. IEEE Press, Los Alamitos (2007)
12. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evolutionary Computation* 9(3), 303–317 (2005)
13. Matthews, K.B., Craw, S., Elder, S., Sibbald, A.R., MacKenzie, I.: Applying genetic algorithms to multi-objective land use planning. In: *Proceedings Genetic and Evolutionary Computation Conference (GECCO 2000)*, pp. 613–620. Morgan Kaufmann, San Francisco (2000)
14. Prell, C., Reed, M.S., Hubacek, K.: Social network analysis and stakeholder analysis for natural resource management. *Society and Natural Resources* (forthcoming, 2007)
15. Termansen, M., McClean, C.J., Preston, C.D.: The use of genetic algorithms and bayesian classification to model species distributions. *Ecological Modelling* 192, 410–424 (2006)
16. Shaw, D.J., Haydon, D.T., Cattadori, I.M., Hudson, P.J., Thirgood, S.J.: The shape of red grouse cycles. *Journal of Animal Ecology* 73, 767–776 (2004)
17. Sotherton, N.W., May, R., Ewald, J., Fletcher, K., Newborn, D.: Managing Uplands for Red Grouse. In: *Managing Uplands for Game and Sporting Interest Conflicts and Changes*. Moors for Future (2007)
18. Dykstra, D.P.: *Mathematical programming for natural resource management*. McGraw-Hill, New York (1984)

19. Hudson, P.J.: Grouse in space and time: the population biology of a managed gamebird. Game Conservancy, Fordingbridge (1992)
20. Liu, J., Dietz, T., Carpenter, S.R., Alberti, M., Folke, C., Moran, E., Pell, A.N., Deadman, P., Kratz, T., Lubchenco, J., Ostrom, E., Ouyang, Z., Provencher, W., Redman, C.L., Schneider, S.H., Taylor, W.W.: Complexity of coupled human-natural systems. *Science* 317, 1513–1516 (2007)
21. Hobbs, R.J.: Length of burning rotation and community composition in high-level Cal-luna-Eriophorum bog in N. England. *Vegetatio* 57, 129–136 (1984)
22. Thompson, D.B.A., MacDonald, A.J., Marsden, J.H., Galbraith, C.A.: Upland heather moorland in Britain: A review of international importance, vegetation change and some objectives for nature conservation. *Biological Conservation* 71, 163–178 (1995)
23. Chapman1, D.S., et al.: Modelling the coupled dynamics of moorland management and vegetation in the UK uplands. *Journal of applied ecology* (to appear)
24. Sotherton, j.N.W., May, R., Ewald, J., Fletcher, K., Newborn, D.: Managing Uplands for Red Grouse. In: *Managing Uplands for Game and Sporting Interest Conflicts and Changes. Moors for Future* (2007)

Evolutionary Money Management

Philip Saks¹ and Dietmar Maringer²

¹ Centre for Computational Finance and Economic Agents,
University of Essex
psaks@essex.ac.uk

² Department for Quantitative Methods, Economics and Business Faculty,
University of Basel
dietmar.maringer@unibas.ch

Abstract. This paper evolves trading strategies using genetic programming on high-frequency tick data of the USD/EUR exchange rate covering the calendar year 2006. This paper proposes a novel quad tree structure for trading system design.

The architecture consists of four trees each solving a separate task, but mutually dependent for overall performance. Specifically, the functions of the trees are related to initiating (“entry”) and terminating (“exit”) long and short positions. Thus, evaluation is contingent on the current market position. Using this architecture the paper investigates the effects of money management. Money management refers to certain measures that traders use to control risk and take profits, but it is found that it has a detrimental effects on performance.

1 Introduction

The foreign exchange (FX) market is the largest financial market in the world. In theory, exchange rates should be intimately linked to macro economic variables, such as interest rates, inflation, money supply and real income. However, in their seminal paper, Meese and Rogoff [11] found that these fundamentals are useless in forecasting exchange rate changes even at medium frequencies. This is known as the *determination puzzle of foreign exchange*. At shorter time horizons many traders tend to use *technical analysis* in decision making [1]. Technical analysis attempts to forecast future price changes based on historical observations. This broad definition covers a wide range of methods from visual pattern recognition to moving averages and more elaborate schemes. Traditionally, it has encountered much skepticism from academia since it clearly contradicts the *Efficient Market Hypothesis* (EMH) – one of the cornerstones of modern finance [6]. Trading on the basis of technical analysis has often been considered irrational behavior, but since it continues to be a widespread approach among practitioners this would have the paradoxical implication that the markets are not efficient to begin with [12]. A basic premise for efficient capital markets is the existence of *homo economicus*.

During the past few decades, there has been an increasing interest in technical analysis among financial economists and extensive literature has emerged on the

subject. There is a general consensus that technical analysis on a daily frequency has been profitable in the past [3, 10, 9]. In contrast, this paper uses high-frequency intraday tick data and considers both a frictionless environment and trading under market frictions through the quoted bid-ask spread. Instead of using a predetermined strategy as a moving average rule or a chart pattern, the computer evolves its own trading strategies using *genetic programming* (GP). GP has previously been applied to trading rule induction for foreign exchange markets, but the results are mixed. In the early nineties GP was found to produce significant profits when trading in the presence of realistic transaction costs [7, 2]. But performance has since deteriorated [5, 13].

As mentioned above, this paper considers high-frequency intraday tick data on the USD/EUR exchange rate covering the full year of 2006. Besides being a much needed update on GP in this domain it adds to the existing literature in a number of ways. The standard approach of GP in trading rule induction is to use a single tree structure that makes buy or sell recommendations. This paper proposes a novel multiple tree structure consisting of four (quad) trees for ternary decision problems. Hence, strategies can take short, neutral and long positions. Which tree is evaluated is contingent on the current market position. Each of the four trees returns Boolean values and their functions can be characterized as long entry, long exit, short entry and short exit. The entry trees initiate either long or short positions, while the exit trees terminate those positions and revert to a neutral state. Using this division of entry and exit strategies, the benefits of money management are examined. Money management refers to certain measures that traders use to control risk and take profits, implying that closing of positions can be initiated by events other than “standard” buy/sell signals. To reflect this in an automated trading system, an extension to standard approaches needs to be made. Traditionally, one common rule for both positions is evaluated, and depending on the outcome, the signal is to enter (stay in) these positions or exit (stay out of) them, respectively. In money management, different rule sets, contingent on the current position, are used. Hence, a negative entry signal is not necessarily seen as an exit signal, but entirely different rules are evaluated to find exit signals. Furthermore, these exit signals can be based on other indicators or information. For example, stop losses are often placed to trigger an exit signal in order to limit downside risk. Since money management is a practitioner’s way of controlling risk, the effects of evolving strategies under different utility functions are investigated.

The paper is outlined as follows. Section 2 presents the data. Section 3 describes the fitness function, model and parameter settings. This is followed by empirical results in Section 4. Finally, Section 5 concludes and gives pointers to possible future research.

2 Data

The data is provided by OANDA FXticks, and comprises of tick data on the USD/EUR exchange rate covering the calendar year 2006. This constitutes a

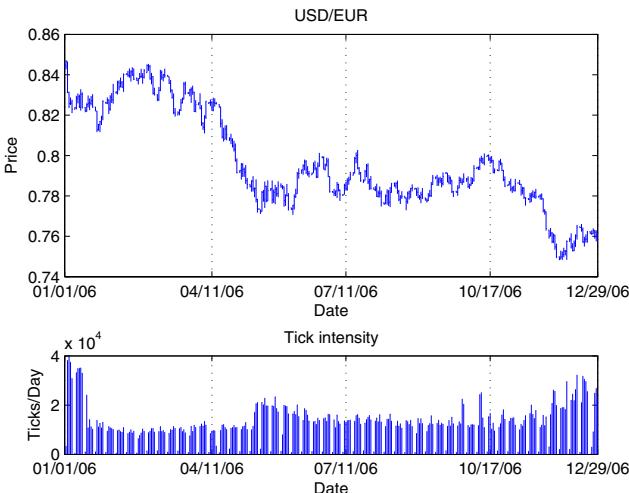


Fig. 1. Daily USD/EUR exchange rate (top), and number of ticks per day (bottom)

Table 1. Summary statistics of log-returns of 10-tick sampled USD/EUR middle prices. AC denotes the auto-correlation and KS is the p -value of a Kolmogorov-Smirnov test.

Mean ($\cdot 10^{-6}$)	Std ($\cdot 10^{-4}$)	Skewness	Kurtosis	AC (lag-1)	AC (lag-2)	KS (p)
-0.277	1.323	-0.215	18.823	-0.048	0.002	0.000

total of 3 894 525 bid-ask observations, and on average there are more than 12 000 per trading day. Figure 1 shows the daily prices, together with the number of ticks per day. Over the entire period the Dollar depreciates from 0.8439 to 0.7577, corresponding to 10.77%. The daily tick intensity appears fairly constant for most of the year, but at the beginning and end of the year the activity is considerably higher. Moreover, during the month of May, there is also increased activity which coincides with a crisis in the equity markets.

The USD/EUR series exhibit strong intraday effects, e.g., the trading activity is higher during European business hours. Moreover, there are two distinct peaks around 08:00 and 14:00 GMT with high activity, separated by lower activity at noon. By using an equidistant intraday sampling in calendar time this seasonality is neglected leading to disproportionately many samples during the night compared to daytime. Hence, sampling is done in trading time. Specifically, the exchange rate is sampled every 10 ticks, which yields 389 452 samples. On average this is close to 1 minute sampling in calendar time. The data comprises bid-ask quotes, but in the following the statistical properties of the logarithmic middle prices are analyzed,

Due to large spikes in the spreads, the top decile is winsorized. For the entire sample, the median spread is 1.1870 bp, with an interquartile range of 0.0707 bp. Table 1 contains summary statistics for the log-returns of the sampled USD/EUR

series. The series has negative skewness and significant excess kurtosis, thus strongly rejecting the null hypothesis of Gaussianity in a Kolmogorov-Smirnov test. At the 10-tick sampling frequency there is a significantly negative first order auto-correlation. This phenomenon has previously been reported in the literature using a one-minute sampling in calendar time [4].

3 Framework

3.1 Objectives and Fitness Function

The choice of objective function is essential in evolutionary computation. Since this paper is concerned with trading system design and ultimately generation of wealth, it is natural to seek inspiration in *economic utility theory*. In classical utility theory, the sole objective of agents is to maximize expected utility of wealth, where more wealth is always preferred to less. However, cognitive psychology has revealed that, in evaluating different outcomes, reference dependence plays a crucial role [8]. In the context of financial investments the reference point is determined by how myopic an agent is, i.e., how frequently wealth is evaluated [17]. This implies that both the long-term level and short-term changes in wealth are important factors in determining overall happiness for speculators. Hence, happiness is a path-dependent function of the evolution of wealth. To capture this path dependency, the period over which the trading rule is optimized is divided into I sub-intervals and a utility is evaluated for each interval. Let the return within an interval i be defined as,

$$r_i = (w_t - w_{t-k})/w_{t-k} \quad (1)$$

where k is the length of the interval. A modified terminal interval wealth is then introduced,

$$\hat{v}_i = \begin{cases} v_0 \cdot (1 + r_i) & \text{if } r_i \geq 0 \\ v_0 \cdot (1 + r_i)^\lambda & \text{if } r_i < 0 \text{ with } \lambda \geq 1 \end{cases} \quad (2)$$

where $\lambda > 1$ implies loss aversion, i.e., a greater sensitivity to decreases in wealth, while equality models no additional disutility of losses beyond risk aversion. The initial endowment at the beginning of the interval is v_0 , but in this paper a unit investor is considered such that $v_0 = 1$. Assuming a power utility function, the value of an interval is,

$$U(\hat{v}_i) = \begin{cases} \frac{\hat{v}_i^{1-\gamma}}{1-\gamma} - \frac{1}{1-\gamma} & \text{if } \gamma > 1 \\ \ln(\hat{v}_i) & \text{if } \gamma = 1 \end{cases} \quad (3)$$

For a risk averse trader, $\gamma > 1$, whereas equality implies risk neutrality. γ controls the concavity of the utility function, and λ regulates the loss aversion by decreasing utilities for losses while leaving utilities for gains unchanged. From this the objective function simply follows as the average interval utility,

$$F = \frac{1}{I} \sum_{i=1}^I U(\hat{v}_i). \quad (4)$$

Table 2. Transition table of the quad tree structure consisting of long entry (T1), long exit (T2), short entry (T3) and short exit (T4), going from the current position to neutral (N), short (S), long (L) position. 0=FALSE, 1=TRUE and -=not evaluated.

	Current position											
	Neutral				Long				Short			
T1	0	0	1	1	-	-	-	-	0	0	1	1
T2	-	-	-	-	0	0	1	1	-	-	-	-
T3	0	1	0	1	0	1	0	1	-	-	-	-
T4	-	-	-	-	-	-	-	-	0	1	0	1
new position	N	S	L	N	L	S	N	S	S	N	L	L

3.2 Model

The purpose of this paper is to evolve trading models using genetic programming. The traditional approach in the context of FX forecasting is to evolve binary decision rules where outputs correspond to either long or short positions in a given currency. Using this representation a trading model is forced to take a directional view and cannot remain neutral. To overcome this problem in a single tree framework, it is possible to construct programs that return a trinary Boolean variable instead of the normal binary Boolean variable [2].

In the context of binary trading models it has previously been found that using a dual tree structure instead of the traditional single tree model has significant impact on performance, especially when market frictions are taken into account. [16]. Capitalizing on these findings this paper proposes a unique quad tree structure. The four trees consist of a long entry (T1), long exit (T2), short entry (T3) and short exit (T4). Unlike a stock, an exchange rate does not have a distinct up and down. The inverse relationship of exchange rates dictates that up for one currency is down for the other and vice versa. In this paper the positions relate to Dollar. Thus, when a long (short) position is initiated we expect an appreciation (depreciation) of the Dollar relative to the Euro.

The workings of the four trees is illustrated in Table 2. Each tree returns a Boolean variable, but which tree is evaluated depends on the current market position. Some cases deserve further explanation. When T1 and T3 are both true, the signal is ambiguous and a neutral position is maintained. Moreover, when both entry and exit signals are true, the strongest views are given precedence such that directional views trump neutrality.

One objective of this paper is to examine the effects of money management on trading strategies. This is done by comparing strategies with special grammars for the exit strategies (T2 and T4), to strategies where the grammar is the same across all the trees. In addition to type constraints the trees have semantic restrictions, which improves the search efficiency significantly, since computational resources are not wasted on nonsensical solutions [2]. The function set for the entry strategies (T1 and T3) consists of numeric comparators, Boolean operators

Table 3. Entry strategy grammar. BTWN checks if the first argument is *between* the second and third. HASINC (HASDEC) returns true if the price has increased (decreased) over the last period.

Function	Arguments	Return Type
+	(price, pConst)	priceNew
<=, >=	(price, price)	bool
<=, >=	(price, priceNew)	bool
<=, >=	(time, timeConst)	bool
BTWN	(price, price, price)	bool
BTWN	(price, priceNew, priceNew)	bool
BTWN	(time, timeConst, timeConst)	bool
HASDEC, HASINC	(lag, price)	bool
AND, OR, XOR	(bool, bool)	bool
NOT	(bool)	bool

Table 4. Additional exit strategy grammar. The complete grammar is composed of the entry strategy grammar and the functions in this table.

Function	Arguments	Return Type
<=, >=	(duration, durationConst)	bool
<=, >=	(profit, pConst)	bool
>=	(drawdown, drawdownConst)	bool
BTWN	(duration, durationConst, durationConst)	bool
BTWN	(profit, pConst, pConst)	bool

and addition. Furthermore, three special functions have been introduced. BTWN takes three arguments and evaluates if the first is between the second and third. HASINC (HASDEC) returns true if the second argument has increased (decreased) over the lag period given by the first argument. The terminals include the variables price and moving averages thereof (price) and the time of day (time). Special constants are available for conditioning on time (timeConst), and the difference between price indicators (pConst). The entry strategy grammar is documented in Table 3.

In practice, traders employ various exit strategies for money management, such as stop losses and profit targets. A stop loss automatically exits the strategy when the current profit is below a certain level. Likewise a profit target closes out a position when a given profit is obtained. A more elaborate scheme is a trailing stop, which ensures that the drawdown does not exceed a given value. Simple stop losses and profit targets might be augmented with time exits such that the duration of a trade is constrained. To capture these ideas the exit strategy grammar contains information about the current profit, drawdown and duration of a trade. The exit strategy grammar is an extension to the entry strategy grammar and their difference is documented in Table 4.

3.3 Parameter Settings

In the following computational experiments a population of 500 individuals is initialized using the *ramped half-and-half* method. It evolves for a maximum of 50 generations, but is stopped after 20 generations if no new elitist (*best-so-far*) individual has been found. A normal tournament selection is used with a size of 5, and the crossover and mutation probabilities are 0.9 and 0.05, respectively. Moreover, the probability of selecting a function node during reproduction is 0.5, and each of the trees in the programs are constrained to a maximum complexity of 25 nodes. Again, this constraint is imposed to minimize the risk of overfitting, but also to facilitate interpretability. If the models lose tractability, it defies the purpose of genetic programming as a knowledge discovery tool.

The entire data set is split into four equal-size blocks of 97,364 samples each. Henceforth the blocks are denoted; I, II, III and IV. Since the blocks have been constructed in trading time, their durations in calendar time differ. Block I covers the period from 01-Jan-2006 to 11-Apr-2006, Block II continues to 11-Jul-2006, Block III ends 17-Oct-2006, and Block IV is the remainder until 29-Dec-2006. The four blocks are shown in Figure 1. The returns in each of the blocks are -2.37%, -4.75%, 1.40% and -5.06%, respectively. In the following experiments, rolling window estimation is made on blocks I-III and successive out-of-sample tests are made on blocks II-IV.

As fitness functions, three different utility functions are considered: risk neutral ($\gamma = 1, \lambda = 1$), risk averse ($\gamma = 35, \lambda = 1$) and loss averse ($\gamma = 35, \lambda = 1.15$). Each Block is divided into a number of sub-intervals, each consisting of 1000 samples. This implies that, on average, wealth is evaluated between one and two times per day, which does not seem unreasonable for a high-frequency trader. As mentioned in Section 3.1, the fitness of an individual trading strategy is the average utility obtained within each sub-interval. Due to the high-frequency domain considered in this paper the overnight interest rates are neglected when calculating the returns of the strategies.

4 Empirical Results

In this section the results from rolling window estimation and testing on blocks I to IV are presented. For statistical inferences ten independent runs are considered for both the entry-entry and entry-exit grammar strategies. Moreover, strategies are evolved both under the assumption of frictionless trading and in the presence of market frictions, i.e., trading occurs on middle prices or through the bid-ask spread, respectively.

The median in-sample fitness value is 60.44 for Block I, and out-of-sample it decreases to 39.05 on Block II. However, when Block II is used for training, the median fitness is identical, which indicates that the strategies have captured the underlying dynamics of the price process perfectly. For the remainder of the blocks the performance is comparable. Overall the interval returns appear relatively normal, albeit with some excess kurtosis in some of the blocks. Under frictionless trading the evolved strategies simply capture the mean-reversion effect

Table 5. Rank sum test p -values for the null hypothesis of equal median fitnesses of the entry-entry and entry-exit grammar strategies

Utility function	In-sample			Out-of-sample		
	I	II	III	II	III	IV
Risk neutral	0.3847	0.6232	0.2730	0.5708	0.9097	0.0312
Risk averse	0.6776	0.7913	0.3447	0.0757	0.4727	0.9097
Loss averse	0.7337	0.4274	0.2413	0.5452	0.9097	0.2413

inherent in the data as documented in Section 2. Exploiting this microstructural effect is a winning strategy regardless of the utility functions considered in this paper. This is true for both the entry-entry and entry-exit grammar strategies. The strategies are generally neutral less than 6% of the time, and the proportion of winning trades is approximately 70%. Given the high turnover it is not surprising that the average return of trades are tiny (≤ 0.1 bp).

Naturally, the introduction of market frictions has an adverse effect on performance. Under risk neutrality the in-sample median fitness in Block I drops to 7.78 and 8.08 for the entry-entry and entry-exit grammar strategies, respectively. What is more interesting is that the out-of-sample median fitnesses in Block II are positive for both grammars (4.51 and 4.37). Unfortunately, this pattern does not repeat itself during the later periods. This holds for all utility functions. However, it should be noted that negative utility need not imply unprofitability. When risk aversion and loss aversion are introduced it has an adverse effect on in-sample performance for both grammars as expected. Moreover, it becomes more difficult to find trading opportunities and the strategies remain neutral a greater proportion of time. Under loss aversion the strategies are neutral 70% of the time.

To test the null hypothesis that the evolved strategies have uncovered significant regularities, their fitness values are compared to that of 1000 randomly initialized strategies. The vast majority of strategies outperform the random strategies at the 0.05 level of significance. For the sake of brevity, detailed results are given elsewhere [15].

As mentioned previously, a main objective of this paper is to examine the effects of money management for different types of speculators. To test formally whether money management, i.e., an extended exit grammar, makes a difference, the Wilcoxon rank sum test is employed for each block and for each utility function. Table 5 lists the p -values for the null hypothesis of identical median fitnesses for the two grammars. Only Block IV out-of-sample under risk neutrality is significant at the usual level. In the context of the other results, this can clearly be treated as a spurious rejection and does not lead to an overall rejection of the null hypothesis. It must therefore be concluded that money management has a detrimental effect on utility, since the evolved strategies do not make use of it. Osler [14] offer a possible explanation for this result: in the foreign exchange markets, stop and limit orders tend to be clustered around round numbers giving rise to distinct support and resistance levels, where trend reversals are more likely to occur. This has not been taken into account in this paper.

5 Conclusion

This paper evolves trading strategies using genetic programming (GP) on high-frequency tick FX data. Furthermore, it proposes a novel quad tree structure for trading system design to allow for a money management system where exit rules can be based on additional indicators and triggers than rules for entering positions.

In practice traders often use so-called money management that builds on a different information set when deciding on whether to exit a trade. This paper makes a distinction between entry and exit strategies, thus providing a more accurate description of the decision problem facing real traders. Evolving money management as an endogenous feature has not previously been attempted in the literature.

The trading strategies are evolved using a fitness measure based on the power utility function, where three different kinds of behavior are investigated: risk neutral, risk averse and loss averse. Evolution is done with and without accounting for transaction costs. In a frictionless environment the strategies exploit the significant mean reverting properties of the returns series. This is a dominating strategy regardless of the utility function, and it proves that the framework is capable of capturing a well-known regularity.

Under market frictions and loss aversion, the strategies spend considerably more time in a neutral position since there are fewer satisfying opportunities. The downside is that generalization can suffer as a result. However, if the patterns discovered in-sample have poor quality, then loss aversion is beneficial because it promotes cautious trading that limits transaction costs.

When comparing the entry-entry and entry-exit grammar strategies, the null hypothesis of identical median performance is not rejected, neither in-sample nor out-of-sample. Hence, the results are not significantly different. This suggests that money management has a detrimental effect on utility, and raises the question as to why it is extensively used by practitioners. A possible explanation is that stop orders and limit orders, in the foreign exchange market, tend to be clustered around round numbers, thus giving rise to distinct support and resistance levels where trend reversals are more likely to occur [14]. This has not been taken into account in this paper, but could be an interesting avenue for future research.

References

- [1] Allen, H., Taylor, M.P.: Charts, noise and fundamentals in the London foreign exchange market. *The Economic Journal* 100(400), 49–59 (1990)
- [2] Bhattacharyya, S., Pictet, O.V., Zumbach, G.: Knowledge-intensive genetic discovery in foreign exchange markets. *IEEE Transactions on Evolutionary Computation* 6(2), 169–181 (2002)
- [3] Chang, K., Osler, C.L.: Methodical madness: Technical analysis and the irrationality of exchange-rate forecasts. *The Economic Journal* 109, 636–661 (1999)

- [4] Dacorogna, M.M., Gencay, R., Müller, U.A., Olsen, R.B., Pictet, O.V.: An Introduction to High-Frequency Finance. Academic Press, London (2001)
- [5] Dempster, M.A.H., Jones, C.M.: A real-time adaptive trading system using genetic programming. *Quantitative Finance* 1, 397–413 (2001)
- [6] Fama, E.F.: Efficient capital markets: A review of theory and empirical work. *Journal of Finance* 25(2), 383–417 (1970)
- [7] Jonsson, H., Madjidi, P., Nordahl, M.G.: Evolution of trading rules for the FX market or how to make money out of GP, Technical report, Institute of Theoretical Physics, Chalmers University of Technology (1997)
- [8] Kahneman, D., Tversky, A.: Prospect theory: An analysis of decision under risk. *Econometrica* 47(2), 263–291 (1979)
- [9] LeBaron, B.: Technical trading profitability in foreign exchange markets in the 1990's, Technical report, Brandeis University (2002)
- [10] Maillet, B., Michel, T.: Further insights on the puzzle of technical analysis profitability. *The European Journal of Finance* 6, 196–224 (2000)
- [11] Meese, R., Rogoff, K.: Empirical exchange rate models of the seventies, do they fit out-of-sample? *Journal of International Economics* 14, 3–24 (1983)
- [12] Menkhoff, L., Taylor, M.P.: The obstinate of foreign exchange professionals: Technical analysis (2006)
- [13] Neely, C.J., Weller, P.A.: Intraday technical trading in the foreign exchange market, Technical report, Federal Reserve Bank of St Louis (1999)
- [14] Osler, C.L.: Currency orders and exchange rate dynamics: An explanation for the predictive success of technical analysis. *The Journal of Finance* 58(5), 1791–1819 (2003)
- [15] Saks, P., Maringer, D.: Evolutionary money management, Technical report, Centre for Computational Finance and Economic Agents, University of Essex (2008)
- [16] Saks, P., Maringer, D.: Genetic programming in statistical arbitrage. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., McCormack, J., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Uyar, A.S., Yang, S. (eds.) *EvoWorkshops 2008. LNCS*, vol. 4974, pp. 73–82. Springer, Heidelberg (2008)
- [17] Thaler, R., Tversky, A., Kahneman, D., Schwartz, A.: The effect of myopia and loss aversion on risk taking: An experimental test. *The Quarterly Journal of Economics* 112(2), 647–661 (1997)

Prediction of Interday Stock Prices Using Developmental and Linear Genetic Programming

Garnett Wilson and Wolfgang Banzhaf

Memorial University of Newfoundland, St. John's, NL, Canada
`{gwilson, banzhaf}@cs.mun.ca`

Abstract. A developmental co-evolutionary genetic programming approach (PAM DGP) is compared to a standard linear genetic programming (LGP) implementation for trading of stocks across market sectors. Both implementations were found to be impressively robust to market fluctuations while reacting efficiently to opportunities for profit, where PAM DGP proved slightly more reactive to market changes than LGP. PAM DGP outperformed, or was competitive with, LGP for all stocks tested. Both implementations had very impressive accuracy in choosing both profitable buy trades and sells that prevented losses, where this occurred in the context of moderately active trading for all stocks. The algorithms also appropriately maintained maximal investment in order to profit from sustained market upswings.

Keywords: Developmental Genetic Programming, Linear Genetic Programming, Computational Finance.

1 Introduction

Technical analysis of the stock market involves attempts to examine the past effects of market movements in order to anticipate what traders will do next to affect the market. Such analysis involves the use of technical indicators to examine price trends and trading volume in order to identify the likely future trading activity and change in price of an asset [1]. In recent years, a number of Evolutionary Computation-inspired algorithms, including genetic programming (GP), have been applied to the analysis of financial markets with a reassuring degree of success [1]. This paper explores the use of a developmental GP system, Probabilistic Adaptive Mapping Developmental Genetic Programming (PAM DGP), that uses co-operative co-evolution of genotype solutions and genotype-phenotype mappings, as well as Linear Genetic Programming (LGP), for interday stock trading. While the encoding of functions is static for LGP, PAM DGP allows emphasis of particular functions over others.

The following section describes previous related approaches to market analysis. Section 3 describes the LGP and PAM DGP implementations and their application to interday trading of individual stocks, as well as the function set for this domain and the related interpretation of an individual's genotype. Results are provided in Section 4 for the analysis of four individual stocks in three market sectors. The conclusions and future work follow in Section 5.

2 Related Approaches to Stock Prediction

Genetic programming approaches have met with considerable success when applied to stock analysis. Yan et al. have shown standard GP to outperform other machine learning techniques such as support vector machines for application to portfolio optimization in highly volatile markets, where this success is attributed to adaptation to optimize profits rather than simply predict returns [2]. Furthermore, the authors found that GP was superior in its balance of Return On Investment (ROI) and robustness to volatility. LGP has been applied to market analysis previously by Grosnan et al. [3], where Nasdaq and Nifty indices were examined. Multi-expression programming (MEP), LGP, and an MEP / LGP ensemble were found to surpass the predictive performance of neural networks or neuro-fuzzy implementations for next day prediction of stock prices. The PAM DGP algorithm that is used in this study relies on a co-evolutionary mechanism. A co-evolutionary process has also been applied to the creation of trading rules by Dreżewski and Sepielak [4] where one species represented entry strategies and one species represented exit strategies. In addition, a multi-agent version of the co-evolutionary algorithm and evolutionary algorithm were tried. For the particular data set used by the authors, the multil-agent co-evolutionary approach generated the most profit. To the authors' knowledge, developmental GP has not been applied to market analysis until this work.

In terms of the application of the GP algorithm to inter-day trading rule generation, a technique somewhat similar to the grammatical evolution (GE) approach of Brabazon and O'Neill [1] was adopted: After a period of initial training, the best evolved rules in the population were used to trade live for a window of n days. The window is then shifted ahead and the current population is retrained on the data within the window on which it was previously trading live in order to trade live on the following n days, and so on. The authors compare two versions of the GE system, one that maintains its population across window-based training periods and one that re-initializes the population with each window shift / training period. The authors found that maintaining the populations, rather than re-initializing them with each window, provided better trading rules that yielded greater profits. As detailed in the following section, our technique uses a shifting window of length 5 days, but shifts only in increments of 1 day. Following the findings and recommendations of [1], populations are not re-evolved with the shifting of each window.

3 PAM DGP and LGP Algorithms for Stock Analysis

In PAM DGP [5], there is a population of genotypes that cooperatively coevolves with a separate population of mappings. A probability table is updated throughout algorithm execution with entries corresponding to each pair of individual genotype and mapping from both populations. The table entries represent frequencies that dictate the probability that roulette selection in a steady state tournament will choose the genotype-phenotype pairing of individuals determined by the indices of the table. The genotype and mapping individual that are members of the current best genotype-mapping pairing are immune to mutation and crossover to maintain the current best solution discovered. Each tournament round involves the selection of four unique

genotype-mapping pairings. Following fitness evaluation and ranking, the probability table columns associated with the winning combinations have the winning combination in that column updated using Equation 1 and the remaining combinations in that column updated using Equation 2

$$P(g, m)_{new} = P(g, m)_{old} + \alpha(1 - P(g, m)_{old}) \quad (1)$$

$$P(g, m)_{new} = P(g, m)_{old} - \alpha(P(g, m)_{old}) \quad (2)$$

where g is the genotype individual / index, m is the mapping individual / index, α is the learning rate (corresponding to how much emphasis is placed on current values versus previous search), and $P(g, m)$ is the probability in table element $[g, m]$. To prevent premature convergence, the algorithm uses a noise threshold. If an element in the table exceeds the noise threshold following a tournament round, a standard Gaussian probability in the interval $[0, 1]$ is placed in that element and all values in its column are re-normalized so the column elements sum to unity. The PAM DGP algorithm and selection mechanism are summarized in Figure 1.

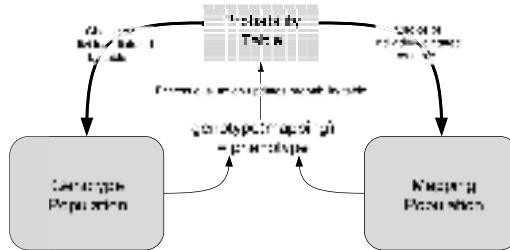


Fig. 1. Probabilistic Adaptive Mapping Developmental Genetic Programming (PAM DGP)

Genotypes in PAM DGP are binary strings, with interpretation of sections of the binary string being instruction-dependent (see next Section 4). Mappings in this work are redundant such that individuals are composed of $b \geq s$ 10-bit binary strings, where b is the minimum number of binary sequences required to represent a function set of s symbols. Each 10 bit mapping section is interpreted as its decimal equivalent, normalized to the range $[0\dots1]$, and mapped to an ordered function set index by multiplying by s and truncating to an integer value (allowing redundant encoding of symbols). Using this mapping mechanism with co-evolutionary selection, PAM DGP will emphasize the most useful members of the function set, ignore members of the function set which are not pertinent, and simultaneously evolve an appropriate genotype solution. PAM DGP is compared to a standard LGP implementation [6]. LGP individuals are also bit strings, and there is naturally only a genotype population. The interpretation of instructions is the same for LGP, using a static mapping and constant function set. PAM DGP extends LGP such that members of a function set are adaptively emphasized. Additional details of PAM DGP are available in [5].

Each steady state tournament consists of 1000 rounds (4 individuals per round). PAM DGP uses a genotype population of size 10 (as does LGP) and mapping population of size 10. Each genotype consists of 320 bits and 4 subresult registers, and each

mapping consists of 160 bits (10 bits for each of 16 required encodings for a function set of size 16). XOR mutation on a (uniform) randomly chosen instruction was used on genotypes, with low threshold point mutation used on mappings to provide a more stable context against which the genotype could evolve. The genotype population used a mutation rate of 0.5 and a crossover rate of 0.9. The mapping population uses a lower crossover and mutation rate, both set at 0.1. PAM DGP used a conservative learning rate of 0.1 and noise threshold of 0.95 to prevent premature convergence.

The PAM DGP and LGP implementations are applied to four stocks, two in the technology sector (Google Inc., ticker symbol “NASDAQ:GOOG”, and Apple Inc., ticker symbol “NASDAQ:AAPL”), one in the energy sector (Chevron Co., ticker symbol “NYSE:CVX”), and one in the consumer/non-cyclical sector (PepsiCo Inc., ticker symbol “NYSE:PEP”). The initial exchange portion of the ticker symbols will be removed for brevity in the remainder of the paper. High, low, open, and close data was provided as input for 200 day periods from 2007 to 2008, with periods chosen to test against performance against particular stock price trends. The first 16 days of the 200 days were reserved as a basis on which to draw technical indicator data. After the first 16 days, the GP fitness was evaluated on data corresponding to a moving window of 5 days. Individuals represent sets of trading rules, based on functions in the function set (to be described). Daily data used for the calculation of a trading decision were normalized using two-phase preprocessing similar to [1]: All daily values were transformed by division by a lagged moving average, and then normalized using linear scaling into the range [0, 1] using

$$v_{scaled} = \frac{v_t - l_n}{h_n - l_n} \quad (3)$$

where v_{scaled} is the normalized daily trading value, v_t is the transformed daily trading value at time step t , h_n is highest transformed value in the last n time steps, l_n is the lowest transformed value in the last n time steps, and n is length of the time lag chosen for the initial transformation ($n = 16$ in this study).

In addition to an instruction set, each individual consists of a set of four registers, a flag for storing the current value of logical operations, and a separate output (trade) register for storing a final value corresponding to a trade recommendation. Following the execution of the trading rules of a GP individual, if the value of the trade register is 0, no action is recommended. Otherwise, the final value in the trade register corresponds to a value in the range [0, 1]. This value was multiplied by a maximum dollar amount to be bought or sold per trade (\$10 000 was used here based on an initial account balance of \$100 000 with which to trade) to give some portion of \$10 000 to be traded. For each trade conducted, there is a \$10 commission penalty. The trading system is permitted to run a small deficit $\geq \$10$ to either handle a sell recommendation when maximally invested (where the deficit would be immediately recouped) or, similarly, to allow a buy in order to be maximally invested. Fitness of an individual is the value of the cash and shares held.

The best individual consisting of the best trading rule set is used by a “live” trading algorithm. That is, the live trader provides known information to the GP for days m to n in the moving window. The GP algorithm returns a recommendation on which the live trading system bases its decision to trade on the following day, $n + 1$. In particular, the net number of shares bought and sold by the best evolved individual given the

recommendation of the trade register over all cases in the sliding window is the buy or sell recommendation to the “live” trading system. With the next window shift, the current cash and shares of stock held by the “live” trading system are the new initial amounts for the GP individuals in the next tournament on the new window content. The transactions of the live trading system are what are actually based on unknown data, and determine the success of the algorithms. The process is summarized in Figure 2.

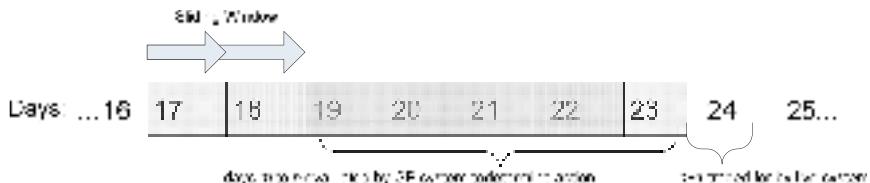


Fig. 2. Relationship between “live” trading system and GP tournament execution

While PAM DGP uses co-evolution to refine function set composition, the appropriate initial function set members must be provided as a basis upon which the algorithm can select its optimum function set. In the case of standard GP, this initial function set remains constant throughout execution. The function set includes standard mathematical operators ($+$, $-$, $*$) and instructions to trade based on logical operators ($<$, $>$, $=$) applied to the four internal registers. In addition, there are established financial analysis metrics of moving average, momentum, channel breakout, and current day high, low, open, or close price. The financial technical indicator *moving average* is the mean of the previous n share prices. The *momentum indicator* provides the rate of change indicator, and is the ratio of a particular time-lagged price to the current price. Momentum is used to measure the strength of the trend of a stock price, and is often used to predict price peaks [1]. *Channel breakout* establishes a trading range for a stock, and reflects its volatility. The most popular solution places Bollinger bands around a n -day moving average of the price at ± 2 standard deviations of the price movement over the last n days (used here with $n = 20$). A trader is typically alerted when the stock price passes the upper or lower bound of the Bollinger bands.

4 Results

The worth of the assets held by the live trading system for each of 184 days of trading is initially analyzed (200 fitness cases were used overall, with the initial 16 being reserved so initial technical financial indicators had values). Fifty such trials over 184 days of trading were conducted for each of the four stocks using an Apple iMac Intel Core 2 Duo 2.8 GHz CPU and 4GB RAM using OS X Leopard v10.5.4. Starting trading with \$100,000, the mean worth (with standard error) of the live trading system for PAM DGP, LGP, and naïve buy-and-hold strategies is given in Figure 3.

Given Figure 3, the prevalent observation is that PAM DGP and LGP are both impressively robust to share price fluctuations (as indicated by the buy and hold trend

line). The evolved solutions seem to take advantage of the upward trends, although the solutions reflect a conservative strategy overall, adept at anticipating and buffering against sharp share price declines and volatility in general. In the instance of PEP, both algorithms are naturally not maximally invested prior to the large market upswing, and thus achieve less final profit (but are still competitive). Both algorithms achieve final profits better than buy-and-hold for the remaining three stocks (GOOG, AAPL, CVX). Figure 4 provides a ratio of PAM DGP to LGP total worth for a finer comparison, with profit (final and cumulative measures) shown in Figure 5.

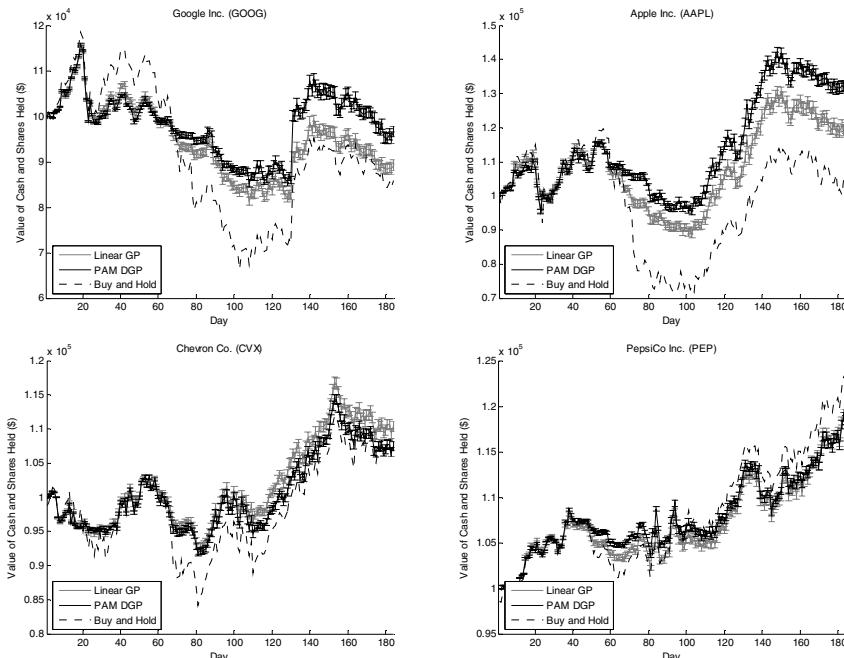


Fig. 3. Mean total worth (value of cash and shares) for PAM DGP, LGP, and buy-and-hold strategies over 50 trials with standard error given initial \$100,000 cash value

Comparing the ratio of PAM DGP and LGP worth across stocks in Figure 4, PAM DGP maintains higher worth than LGP for the large majority of trading days in the instances of GOOG, AAPL, and PEP, with LGP dominating PAM DGP almost the entire period for CVX. PAM DGP outperforms LGP by over 10% at times for GOOG and AAPL, but when LGP outperforms PAM DGP throughout CVX it is by a lower margin (just over 5%). Comparing Figures 3 and 4, it is evident that PAM DGP provides increased robustness to market downturns and quickly takes advantage of growth opportunities later in evolution. Also, Figure 3 and 4 indicate that LGP slightly outperforms PAM DGP for CVX by not selling quite as much stock during a market dip immediately preceding a steady climb starting at approximately day 100.

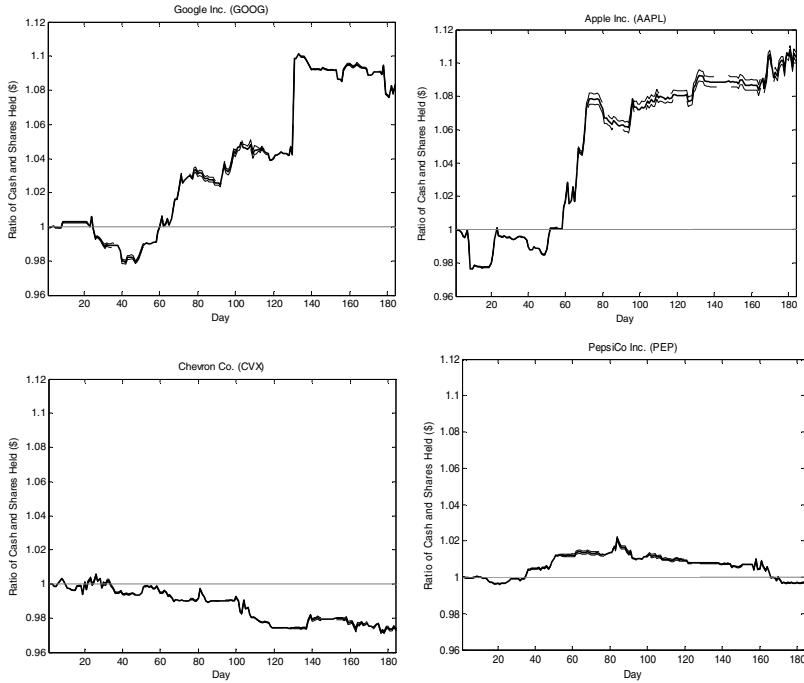


Fig. 4. Mean ratio (solid line) of PAM DGP to LGP live trading system total worth over 50 trials with standard error (dotted line). Values greater than 1 indicate greater PAM DGP worth than LGP, values less than 1 *vice versa*.

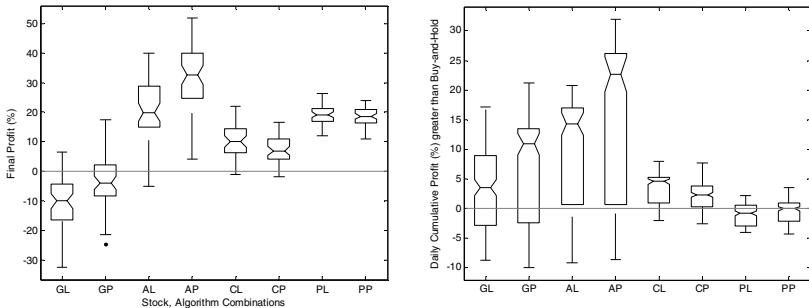


Fig. 5. Boxplot of mean final profit (%) and mean daily cumulative profit (%) greater than buy-and-hold for PAM DGP and LGP over 50 trials. First letter of label indicates stock, second letter indicates algorithm. Value of 0 indicates the break even point.

In the boxplots of Figure 5, each box indicates the lower quartile, median, and upper quartile values. If the notches of two boxes do not overlap, the medians of the two groups differ at the 0.95 confidence interval. Points represent outliers to whiskers of 1.5 times the interquartile range. PAM DGP outperforms LGP at the end of the time

period (Figure 5, left) for GOOG and AAPL, with LGP slightly outperforming PAM DGP for CVX, and no statistically significant difference in final profits for PEP (all at the 95% confidence interval). Figure 5 (left) also shows impressive final profit for AAPL, PEP, and CVX. There was a general loss for both PAM DGP and LGP considering final profit for GOOG. GOOG incurred losses during most of the time period and was thus not profitable overall. Note that time period end is arbitrary and profits are a direct reflection of underlying market trend. Figure 5 (right) shows the mean daily cumulative profit (%) greater than buy-and-hold for the LGP and PAM DGP live trading systems over all trading days. Figure 5 (right) indicates that both PAM DGP and LGP were generally more profitable than buy-and-hold at any given time for all stocks (except, naturally, the case of PEP where naïve buy-and-sell is a very good strategy). PAM DGP was more profitable than LGP at any given time by a large margin for GOOG and AAPL, by a slight margin for PEP, and LGP slightly outperformed PAM DGP for CVX (all at the 95% confidence interval). Number of shares retained daily as a percentage of live trading total worth is shown in Figure 6.

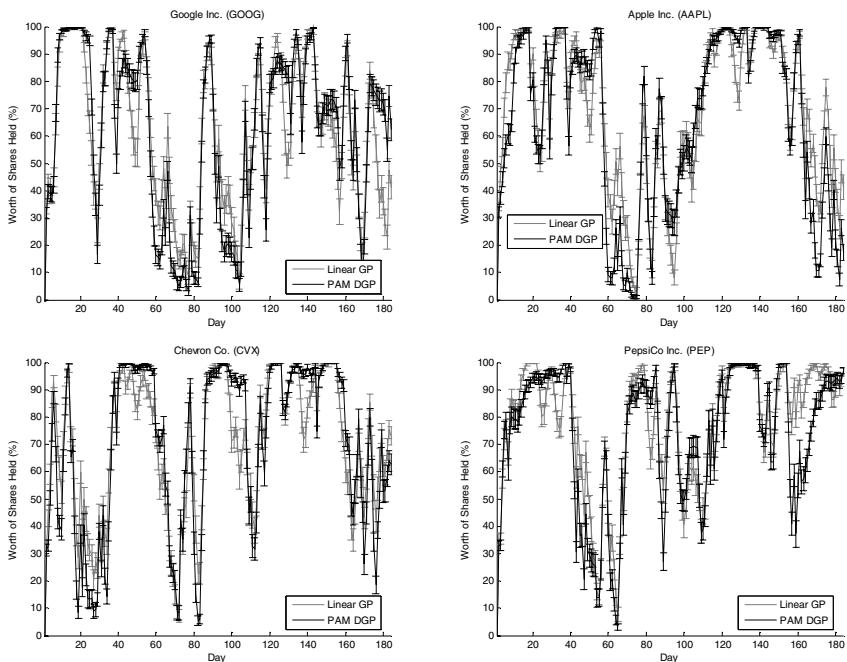


Fig. 6. Mean shares held by PAM DGP (black line) and LGP (grey line) live trading systems as a percentage of total worth over 50 trials with standard error

Comparing Figures 3 and 6, it is evident that both PAM DGP and LGP are impressively reactive in that they will sell stock if a market downturn starts and buy when the market appears to be experiencing gains. Figures 3 and 6 also indicate that both algorithms are effective at staying maximally invested during profitable periods. The allocation of resources in or out of the market is a result of the underlying trading

activity, shown in Figure 7. Proportion of profitable trades is a common metric for evaluation of trading activity, although it is deceptive: it does not even reflect the overall ability of an algorithm in terms of actual profit generated [1]. Many trades, although not profitable, are beneficial in preventing loss during market downturns. Thus, rather than percentage of profitable trades, the percentage of profitable buy trades and percentage of sell trades preventing loss for each algorithm are shown in the top left and right boxplot of Figure 7, respectively. The percentage of trading opportunities where action was taken is shown in Figure 7 (bottom left). Out of all possible trades, the number of trades not conducted when the system was maximally or minimally invested is shown in Figure 7 (bottom right).

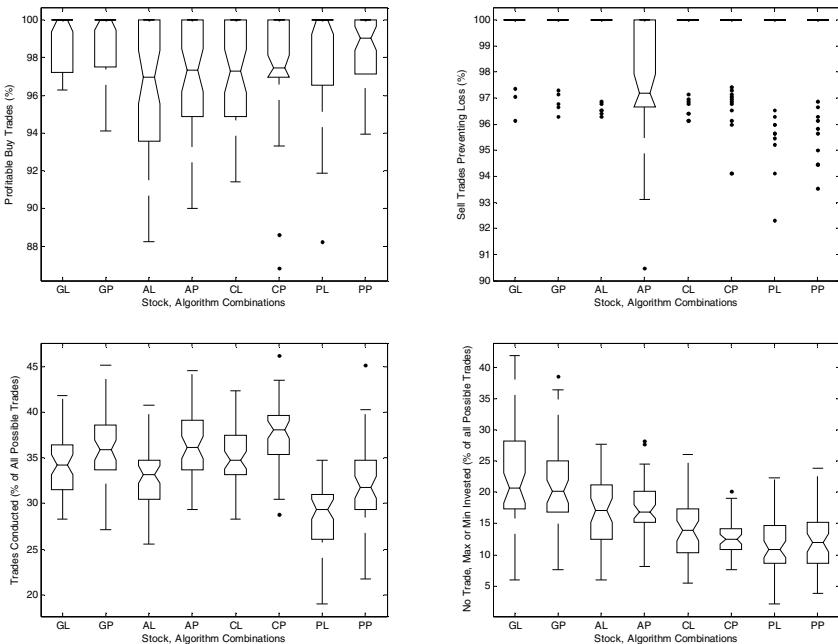


Fig. 7. Percentage of profitable buy trades, sell trades preventing losses, percentage of trades executed overall for each stock, and percentage of trades not conducted while maximally or minimally invested for each algorithm combination over 50 trials. First letter of label indicates stock, second letter indicates algorithm.

Figure 7 reveals that both algorithms are extremely accurate at buying to gain profit, with medians of 96% - 100% successful buys. In terms of protecting investment through selling to prevent loss, the median for both algorithms was typically 100%. There is no statistical difference (at the 95% confidence interval) in the ability of PAM DGP or LGP to buy for profit or sell to prevent loss for any of the stocks examined (with the exception of PAM DGP when selling for AAPL). Any outliers in either buying for profit or selling to prevent loss were acceptably high percentages. These beneficial transactions are also the result of trading levels with medians of 30%

to 40% of possible trades for GOOG, AAPL, and CVX (Figure 7, bottom left), with lower medians for PEP (fewer trades were best to take advantage of the maintained upward trend of PEP). PAM DGP generally conducted more trades (based on spread of data) than LGP for all stocks. Figure 7 (right, bottom) indicates medians between 10% and 22% of trades where the system wished to maintain a maximally or minimally invested position. Compared with Figure 6, it is evident that most of these positions were maximal investment to generate profit. Overall, Figure 7 indicates that the percentage of beneficial trades that were made to generate profit or protect from losses were impressively high, where this occurred in the context of moderate levels of trading.

5 Conclusions and Future Work

This work examined the trading performance of a co-evolutionary developmental GP model (PAM DGP) using a genotype-phenotype mapping and more traditional LGP on four stocks. Both algorithms were robust to share price fluctuations compared to naïve buy-and-hold strategy. Both algorithms efficiently adapted to guard investments during market downturns and readily take advantage of market gains. PAM DGP definitively outperformed standard LGP in final and cumulative profit in 2 of 4 stocks, with slightly better, competitive, or similar results for the remaining 2 stocks. Both algorithms exhibited impressive accuracy in choosing beneficial trades, both for profitable buys and selling to protect investments, and did so with moderate levels of trading and periods of maximal investing to capitalize on market upswings. Future work will examine options for risk adjusted fitness and portfolio management.

References

1. Brabazon, A., O'Neill, M.: *Biologically Inspired Algorithms for Financial Modelling*. Springer, Heidelberg (2006)
2. Yan, W., Sewell, M., Clack, C.D.: Learning to Optimize Profits Beats Predicting Returns — Comparing Techniques for Financial Portfolio Optimisation. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2008, pp. 1681–1688. ACM Press, New York (2008)
3. Grosan, C., Abraham, A.: Stock Market Modeling Using Genetic Programming Ensembles. *Studies in Computational Intelligence* 13, 131–146 (2006)
4. Drezewski, R., Sepielak, J.: Evolutionary System for Generating Investment Strategies. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008*. LNCS, vol. 4974, pp. 83–92. Springer, Heidelberg (2008)
5. Wilson, G., Heywood, M.: Introducing Probabilistic Adaptive Mapping Developmental Genetic Programming with Redundant Mappings. *Genetic Programming and Evolvable Machines* 8, 187–220 (2007)
6. Brameier, M., Banzhaf, W.: *Linear Genetic Programming*. Springer, New York (2007)

An Introduction to Natural Computing in Finance

Jing Dang^{1,2}, Anthony Brabazon^{1,2}, David Edelman^{1,2}, and Michael O'Neill^{1,3}

¹ Natural Computing Research and Applications Group,
University College Dublin, Ireland

² School of Business, University College Dublin, Ireland

³ School of Computer Science and Informatics, University College Dublin, Ireland
jing.dang@ucd.ie, anthony.brabazon@ucd.ie, m.oneill@ucd.ie, davide@ucd.ie

Abstract. The field of Natural Computing (NC) has advanced rapidly over the past decade. One significant offshoot of this progress has been the application of NC methods in finance. This paper provides an introduction to a wide range of financial problems to which NC methods have been usefully applied. The paper also identifies open issues and suggests multiple future directions for the application of NC methods in finance.

1 Introduction

Recent years have seen the application of multiple Natural Computing (NC) algorithms (defined in this paper as computer algorithms whose design draws inspiration from phenomena in the natural world) for the purposes of financial modelling [7]. Particular features of financial markets including their dynamic and interconnected characteristics bear parallel with processes in the natural world and *prima facie*, this makes NC methods ‘interesting’ for financial modelling applications. Another feature of both natural and financial environments is the phenomenon of emergence, or the activities of multiple individual agents combining to co-evolve their own environment.

The scale of NC applications in finance is illustrated by Chen & Kuo[10] who list nearly 400 papers that had been published by 2001 on the use of evolutionary computation alone in computational economics and finance. Since then several hundred additional papers have been published illustrating the continued growth in this application area (see also [53,47,8] for additional examples of NC applications in finance). In this paper we provide a concise review of some of this work, describing the utility of NC methods within each of the following financial areas: forecasting, algorithmic trading, portfolio optimisation, risk management, derivative modelling and agent-based market modelling. As NC methods have general utility as optimisation, model induction and agent-based modelling tools, there is potential for wide application of these methods in finance.

The rest of this paper is organised as follows. Section 2 provides a concise overview of some key families of NC methods. Section 3 introduces various financial applications of NC methods and shows how these methodologies can add value in those applications. Section 4 concludes this paper, suggesting multiple avenues of future work at the intersection of finance and natural computing.

2 Natural Computing

Natural computing (NC) algorithms can be clustered into different groups depending on the aspects of the natural world upon which they are based. The main clusters are illustrated in Fig.1 below.

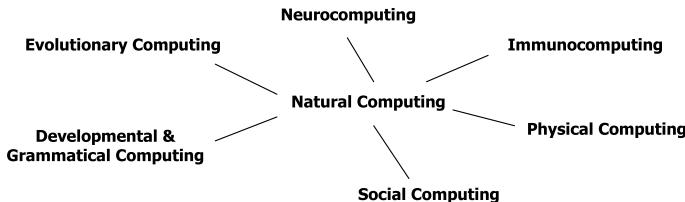


Fig. 1. An overview of the main families of Natural Computing Algorithms

Neurocomputing (or neural networks, NNs) typically draws inspiration from the workings of human brain or nervous system. NNs can be characterised by a set of neurons, the network structure describing the pattern of connectivity between neurons, and the learning approach used. The predominant neurocomputing paradigms include *multi-layer perceptrons*, *radial basis function networks*, *self-organising maps*, and *adaptive resonance theory*.

Evolutionary Computation (EC) is based upon neo-Darwinian principles of evolution. A population-based search process is used, whereby better (fitter) members of the population are selected for reproduction and modification, leading to a new population of individuals increasingly adapted to their environment. The main streams of EC are *genetic algorithms*(GA), *evolution strategies*(ES), *evolutionary programming*(EP) and *genetic programming* (GP).

Social Computing adopts a swarm metaphor and includes algorithms inspired by the flocking and schooling behaviour of birds and fish, and the behaviours observed in social insects such as ants. The characteristics of these social systems facilitate self-organisation, flexibility, robustness, and direct or indirect communication among members of the population. Examples of social computing include *ant colony*, *particle swarm* and *bacterial foraging* algorithms.

Immunocomputing encompasses a family of algorithms inspired by the complex and adaptive biological immune system of vertebrates. The natural immune system represents an intricate network of specialised chemicals, cells, tissues and organs with the ability to recognise, destroy and remember an almost unlimited number of foreign bodies, and to protect the organism from misbehaving cells.

Physical Computing draws inspiration from the physical processes of the natural world, such as simulated annealing and quantum mechanics. A claimed benefit of the quantum-inspired algorithms is that because they use a quantum

representation, they can maintain a good balance between exploration and exploitation, they can also offer computational efficiencies since smaller population sizes can be used compared to typical evolutionary algorithms.

Developmental and Grammatical Computing borrows from both a developmental and a grammar metaphor. Grammatical computing adopts concepts from linguistic grammars, where generative grammars are used to construct a sentence(s) in the language specified by the grammar. This process is metaphorically similar to the developmental process in biology where ‘rules’ govern the production of a multi-cellular organism from a single cell. An example is *grammatical evolution* (GE), which is a grammatical variant of genetic programming.

These families of NC algorithms provide a rich set of tools for the development of quality optimisation, model induction and agent-based modelling applications, and all have seen applications in finance. Readers requiring detailed information on these algorithms are referred to [13,25].

3 Financial Applications

In this section we introduce the application of NC methods across a range of financial areas including forecasting, algorithmic trading, portfolio optimisation, risk management, derivative modelling and agent-based market modelling.

3.1 Forecasting

Financial markets are affected by a myriad of interacting economic, political and social events. The relationships between these factors are not well understood, and moreover, not stationary over time. To predict future values of macroeconomic variables, market indices, the volatility of some financial products, etc., NC methods can be applied for parameter estimation (optimisation), or more generally for model induction wherein the structure of the underlying data generation process is uncovered.

For example, GA has been used to select explanatory variables from financial statements in order to predict corporate earnings [45] and for prediction of IPO underpricing [40]. NNs have been employed for the prediction of index levels [7], take-over targets [21] and auditor qualification of financial statements [44]. GP has been used to forecast exchange rate volatility [39] and for developing investment rules [46].

Typically, studies applying NC methods for financial forecasting use measures of goodness of fit such as mean squared error, mean absolute percentage error etc. as their fitness function. The aim is to uncover or train a model which ‘fits’ a historic dataset well. Unsurprisingly, the choice of fitness function usually has a critical impact on the behaviour of resulting model, hence a model constructed using one fitness measure will not necessarily perform well on another.

3.2 Algorithmic Trading

Algorithmic (or automated) trading is the use of computer programs to make decisions relating to some aspect of the trading of financial instruments including

the timing, price, or even the final quantity of the order. Below we illustrate some important processes of algorithmic trading where NC methods can be applied.

Trading Strategy

A trading strategy is a predefined set of rules for making trading decisions. Algorithmic trading can be applied in any trading strategy such as hedging, arbitrage, or pure speculation (including trend following). A trading rule should specify the entry, profit-taking, and stop-loss or exit strategy. NC methods can be used for rule optimisation (to find optimal parameters for a fixed rule) or rule induction (to find optimal combination of diversified rules). Early applications of EC to uncover trading rules include [6,38,3]. Markets have periods in which a rule can work, but it is hard to find evidence of rules which are successful over long time periods. Hence an adaptive trading strategy seems more plausible [20,12]. Instead of typical data drawn from the market, financial statements or macroeconomic data, [43,30] used text data drawn from either internet message boards or the financial press to find market trends and evolve trading rules. A wide range of hybrid forecasting approaches have also been employed such as the neuro-fuzzy hybrids [54], neuro-genetic hybrids, geno-fuzzy [20] and ensemble methods (combining multiple models to make a trading decision) [29].

Trade Execution

An important issue in trading financial assets is the efficient trade execution. The design of trade execution strategies seeks to balance out the cost of market impact and opportunity cost. In selecting a trade execution strategy, the investor (or the computer if the process is automated) decides the number of slices the trade will be split into, the size of each individual trade, when to submit an order, and the type of order to use (limit or market). Applications of NC for optimal trade execution have begun to emerge in recent years. For example, Lim & Coggins [33] used a GA to evolve a strategy in order to optimise trade execution performance using order book data from the Australian Stock Exchange. In their approach, the basic structure of the execution rule is determined in advance (number of trades etc.) and the task of GA is to parameterise the rule.

3.3 Portfolio Optimisation

'Optimization is the engineering part of portfolio construction', as mentioned by Fabozzi et al [18] in their recent survey for quantitative equity portfolio management: *'Most portfolio construction problems can be cast in an optimization framework, where optimization is applied to obtain the desired optimal risk-return profile'*. Multiple elements of the portfolio management process, such as asset allocation, security selection, index tracking, etc. where optimisation is crucial, are amenable to NC methodologies.

A classical approach used for portfolio optimisation is the Markowitz mean-variance model [36]. It assumes that investors wish to maximise their return (measured as mean or expected return) and minimise their risk (measured as variance or the standard deviation of their return). Real-world portfolio optimisation can present a difficult, high-dimensional, constrained optimisation

problem. In this setting, heuristic approaches such as evolutionary computing methods are of particular interest because of their ability to find good solutions even if optimality is not assured. For example, *differential evolution* (DE) has been applied to the index tracking problem [35] and *evolutionary algorithm* has been used for multi-period pension fund asset allocation problem [5,41]. Practically, portfolio managers are concerned with a variety of risk and return measures, not just expected return and its variance. A variety of papers have applied Multi-Objective Evolutionary Algorithms (MOEAs, see [9] for a review of their applications in finance) to non-Markowitz risk metrics, including [34] which uses a compound risk metric. An evolutionary stochastic portfolio optimisation method is introduced in [22] and applicable to a set of structurally different risk measures. Recent work has also seen the application of coevolutionary MOEAs for portfolio optimisation [16].

3.4 Risk Management

Risk management is a critical aspect of the investment decision. Some of the main types of risks faced by investors are illustrated in Fig. 2 below.



Fig. 2. Risk illustration

Market Risk Computation

Market risk refers to the risk faced by an investor arising from changes in financial market prices. Traditionally, a measure for market risk is Value-at-Risk (VaR), which measures the worst expected loss under normal market conditions over a specific time interval. The loss distribution is usually assumed to be Normal when calculating VaR. Evolutionary algorithms do not need to embed this assumption and can incorporate any preferred loss distribution type [49]. Other market risk measures such as Conditional VaR estimates and expected shortfall can be calculated using NNs [32,15].

Credit Risk Assessment

Credit risk is the risk that the counterparty to a deal fails to perform their obligations. Over the past several decades an extensive literature has amassed on modelling creditworthiness and default risk. The objective is to develop a

model which will provide a metric of creditworthiness from a series of explanatory variables. As the range of NC techniques have expanded over the past twenty years, each new technique has been applied to credit scoring and corporate failure prediction. Examples include feedforward neural networks [4], self-organising maps [42,27], GAs [28,50], Ant models [51,7], GP and GE [37,7,2].

Of course, there are other types of risks which need to be quantified in practice, such as liquidity risk (arising when an institution is unable to raise cash to fund its business activities) and operational risk (arising due to poor or inadequate management control systems or due to human error). However, as yet, there is little literature concerning the application of NC methods in these areas.

3.5 Derivatives Modelling

Derivatives are contracts whose value is derived from the value of the underlying assets, such as equities, interest rates, currencies, market indices, commodities etc.. Two of the best known forms of derivative are futures and options. A future is an agreement to buy or sell goods, currency or securities on an agreed future date and for a price fixed in advance. An option is a financial instrument simply gives the holder (buyer) the right, but not the obligation, to buy or sell a specified underlying asset at a pre-agreed price on or before a given date.

The key issue for investors wishing to trade in derivatives is the determination of the fair price for the derivative. There have been two main avenues of application of NC methods in pricing financial derivatives, namely, model calibration and model induction. In model calibration, the objective is to estimate the parameters of (or ‘calibrate’) a theoretical pricing model. The parameters are estimated by fitting the model to the relevant returns time series. Typically the pricing model will have a complex, non-linear structure with multiple parameters, where global search heuristics such as GA can be utilised to uncover a high-quality set of parameters. Examples of using NC algorithms for model calibration include [14,19]. In model induction, NC approach such as self-organising, fuzzy neural networks or GP would have particular utility when little is known about the underlying asset pricing dynamics as both the structure and the parameters of the pricing model are estimated directly from the data, thereby extracting the pricing model implicitly. Applications of such methods include NNs [23,52,48] or GP [26,11] to recover a proxy for the price-generating model.

3.6 Agent-Based Market Modelling

The essence of Agent-based Modelling (ABM) lies in the notion of autonomous agents whose behaviour evolves endogenously leading to complex, emergent, system dynamics which are not predictable from the properties of individual agents. In designing ABMs of financial markets, NC methods can be used to model the information processing and storage by agents, the process of adaptive learning by agents, or to model the trading mechanism. One example of the use of ABM to simulate a financial market is provided by LeBaron [31]. This model simulates price generation and trading in a market made up of artificial adaptive agents.

Other applications of ABM include the simulation of a foreign exchange market [24], the modelling of artificial stock option market [17] and the modelling of an artificial payment card market [1]. A key output from the ABM literature on financial markets is that it illustrates that complex market behaviour can arise from the interaction of quite simple agents. Carefully constructed, ABM can help increase our understanding of market processes and can potentially provide insights for policy makers and regulators. Of course, issues of model validation are important in all ABM applications including those in financial markets.

4 The Future

Though a plethora of academic literature on NC applications in finance exists, it is notable that many papers have concerned proof of concept rather than robust, industry-strength finance applications. While NC methods offer potential in multiple finance areas, the maturing of their application requires future work focusing on complex real-world finance problems. This will require the construction of multi-disciplinary research teams with possible industrial collaborations, drawing expertise as necessary from finance, computer science, mathematics, biology, etc.. Some promising future directions for research at the nexus of natural computing and finance include:

- ***Forecasting:*** While forecasting models applying NC methods can typically be constructed to fit historical data fairly well, a common finding is that the quality of the out-of-sample forecasts diminishes over time. Hence we can expect to see increased use of more sophisticated methods for pre-processing the raw time-series inputs, and for the adaptation of the resulting models in response to changing environmental conditions. Another area of growing interest is the incorporation of information from text mining (e.g. from the financial press) into forecasting models.
- ***Algorithmic Trading:*** While many published papers have focussed on the development of simplified trading systems, successful real-world applications have often focussed on the support of specific elements of the investment process. In the medium term, we can expect to see a notable increase in the rigor of published work in this area as computer scientists form teams with finance academics and practitioners, incorporating realistic models of market microstructure. The area of trade execution has seen relatively little published application of NC methodologies, despite its real-world significance. Model induction tools such as GP offer interesting potential here, as do agent-based modelling approaches. The latter could be used to uncover robust trade execution strategies.
- ***Portfolio Optimisation:*** There has already been an extensive application of NC methods for portfolio optimisation, but a further systematic investigation is still needed, i.e., dealing with portfolio constraints for special sub-application areas, e.g., hedge fund, pension fund, insurance, etc.; improving the effectiveness and efficiency for dynamic asset allocation, optimising portolio managing strategies taking account of reasonably multiple risk measures.

- **Risk Management:** Recent events on the financial markets have underscored the importance of risk management, and the weakness of existing theoretical models in this area. It is interesting to note that applications of NC methods in risk management have not attracted as much attention as might be expected in the literature and this remains an open research area. For example, NC methods could be applied to model assist development of enterprise-wide risk management, improve the flexibility and efficiency for large scale multi-stage *asset liability management* (ALM) models with millions of variables and constraints.
- **Derivatives Modelling:** In spite of the vast array of derivatives products available, and the weakness of financial theory once we move beyond vanilla products, there have only been a relatively limited number of applications of NC for model calibration or model induction in this area. Possibilities also exist to hybridise NC methods with traditional numerical methods.
- **Agent-based Market Modelling:** The field of ABM is attracting significant attention with the increasing questioning of agent homogeneity which underlies classical financial economics. ABM allows us to examine the effect of differing forms of market structure on market behaviour. Doubtless, the next few years will see increased focus on this given the failures of market regulation over the of the recent financial crisis. The co-evolutionary element of markets lends itself well to NC approaches in terms of modelling of agent behaviour and strategy adaptation.

Some elements affecting above various financial areas, such as market microstructure, liquidity, etc. are also worth to be explored in future research. A practical issue arisen in applications of NC to finance is that the underlying algorithms are themselves undergoing a process of maturation. Recent years have seen extensive research in order to extend the canonical algorithms into high-dimensional problem environments (scalability), to develop the efficient algorithms for constrained optimisation, and to develop practical application of the algorithms in dynamic problem environments. Meantime, we have seen developments in computer hardware, and in our ability to implement parallel versions of NC algorithms (e.g. GPU implementations). These two strands of development are creating an ever more powerful toolbox of NC algorithms for financial modellers. In future, we await a systematic applications of these methods growing from both the natural computing and the computational finance community.

References

1. Alexandrova-Kabadjova, B., Tsang, E., Krause, A.: Evolutionary learning of the optimal pricing strategy in an artificial payment card market. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computing in Computational Finance* (2008)
2. Alfaro-Cid, E., Cuesta-Canada, A., et al.: Strong typing, variable reduction & bloat control for solving the bankruptcy prediction problem using genetic programming. In: *Natural Computing in Computational Finance*. Springer, Heidelberg (2008)
3. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. *Journal of Financial Economics* 51, 245–271 (1999)

4. Atiya, A.: Bankruptcy prediction for credit risk using neural networks: A survey and new results. *IEEE Trans. Neural Networks* 12(4), 929–935 (2001)
5. Baglioni, S., Sorbello, D., Da Costa Pereira, C., Tettamanzi, A.G.B.: Evolutionary multiperiod asset allocation. In: *Proceedings of GECCO 2000*, pp. 597–604 (2000)
6. Bauer, R.: *Genetic Algorithms and Investment Strategies*. Wiley, Chichester (1994)
7. Brabazon, A., O'Neill, M.: *Biologically Inspired Algorithms for Financial Modelling*. Springer, Berlin (2006)
8. Brabazon, A., O'Neill, M. (eds.): *Natural Computing in Computational Finance*. Springer, Berlin (2008)
9. Castillo Tapia, M.G., Coello, C.A.C.: Applications of multi-objective evolutionary algorithms in economics and finance: a survey. In: *Proceedings of CEC 2007*, pp. 532–539. IEEE Press, Los Alamitos (2007)
10. Chen, S.-H. (ed.): *Evolutionary Computation in Economics and Finance*. Physica-Verlag (2002)
11. Chidambaran, N.: Genetic programming with Monte Carlo simulation for option pricing. In: *Proceedings of IEEE Winter Simulation Conference 2003*, pp. 285–292 (2003)
12. da Costa Pereira, C., Tettamanzi, A.: Fuzzy-evolutionary modeling for single-position day trading. In: *Natural Computing in Computational Finance* (2008)
13. de Castro, L.N.: Fundamentals of natural computing: an overview. *Physics of Life Reviews* 4(1), 1–36 (2007)
14. Dang, J., et al.: Option model calibration using a bacterial foraging optimisation algorithm. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008*. LNCS, vol. 4974, pp. 133–143. Springer, Heidelberg (2008)
15. Diagne, M.: Financial risk management and portfolio optimization using artificial neural networks and extreme value theory. Univ. of Kaiserslautern (2002)
16. Dreżewski, R., Siwik, L.: Co-Evolutionary Multi-Agent System for Portfolio Optimization. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computing in Computational Finance*. Springer, Berlin (2008)
17. Ecca, S., Marchesi, M., Setzu, A.: Modeling and simulation of an artificial stock option market. *Computational Economics* 32(1), 37–53 (2008)
18. Fabozzi, F.J., et al.: Trends in quantitative equity management: survey results. *Quantitative Finance* 7(2), 115–122 (2007)
19. Fan, K., et al.: Quantum-inspired evolutionary algorithms for calibration of the VG option pricing model. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) *EvoBIO 2007*. LNCS, vol. 4447, pp. 186–195. Springer, Heidelberg (2007)
20. Ghandar, A., Michalewicz, Z., et al.: Computational Intelligence for Evolving Trading Rules. *IEEE Transactions on Evolutionary Computation* (2008)
21. Hickey, R., Little, E., Brabazon, A.: Identifying merger and takeover targets using a self-organising map. In: *Proceedings of ICAI 2006*. CSEA Press (2006)
22. Hochreiter, H.: Evolutionary stochastic portfolio optimization. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computing in Computational Finance* (2008)
23. Hutchinson, J., Lo, A., et al.: A non-parametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 851–889 (1994)
24. Izumi, K.: An artificial market model of a foreign exchange market, PhD Thesis, Tokyo University (1999)
25. Kari, L., Rozenberg, G.: The many facets of natural computing. *Communications of the ACM* 51(10), 72–83 (2008)
26. Keber, C.: Option valuation with the genetic programming approach. In: *Computational Finance - Proceedings of the sixth international conference*, pp. 689–703. MIT Press, Cambridge (2000)

27. Kiviluoto, K., Bergius, P.: Maps for analysing failures of small and medium-sized enterprises. In: Deboeck, G., Kohonen, T. (eds.) *Visual Explorations in Finance with Self-Organizing Maps*, pp. 59–71. Springer, Berlin (1998)
28. Kumar, N., Krovi, R., Rajagopalan, B.: Financial decision support with hybrid genetic and neural based modeling tools. *European Journal of Operational Research* 103(2), 339–349 (1997)
29. Kwon, Y.-K., Moon, B.-R.: Evolutionary ensemble for stock prediction. In: Deb, K., et al. (eds.) *GECCO 2004. LNCS*, vol. 3103, pp. 1102–1113. Springer, Heidelberg (2004)
30. Larkin, F., Ryan, C.: Good News: Using news feeds with genetic programming to predict stock prices. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) *EuroGP 2008. LNCS*, vol. 4971, pp. 49–60. Springer, Heidelberg (2008)
31. LeBaron, B.: Building the Santa Fe artificial stock market. Working paper, Brandeis University (2002)
32. Lee, H., et al.: Coherent risk measure using feedforward neural networks. In: Wang, J., Liao, X.-F., Yi, Z. (eds.) *ISNN 2005. LNCS*, vol. 3497, pp. 904–909. Springer, Heidelberg (2005)
33. Lim, M., Coggins, R.: Optimal trade execution: An evolutionary approach. In: *Proceedings of CEC 2005*, pp. 1045–1052. IEEE Press, Los Alamitos (2005)
34. Lipinski, P.: Evolutionary strategies for building risk-optimal portfolios. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computing in Computational Finance* (2008)
35. Maringer, D.: Constrained Index Tracking under Loss Aversion Using Differential Evolution. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computing in Computational Finance*. Springer, Berlin (2008)
36. Markowitz, H.: Portfolio Selection. *Journal of Finance* 1(7), 77–91 (1952)
37. McKee, T., Lensberg, T.: Genetic programming and rough sets: a hybrid approach to bankruptcy classification. *European Journal of Operational Research* 138, 436–451 (2002)
38. Neely, C., Weller, P., Dittmar, R.: Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis* 32(4), 405–428 (1997)
39. Neely, C., Weller, P.: Using a genetic program to predict exchange rate volatility. In: Chen, S.-H. (ed.) *Genetic Algorithms and Genetic Programming in Computational Finance*, pp. 263–278. Kluwer Academic Publishers, Dordrecht (2002)
40. Quintana, D., Luque, C., Issasi, P.: Evolutionary rule-based system for IPO underpricing prediction. In: *Proceedings of GECCO 2005*, pp. 983–989. ACM, New York (2005)
41. Senel, K., Pamukcu, A.B., Yanik, S.: An evolutionary approach to asset allocation in defined contribution pension schemes. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computing in Computational Finance*. Springer, Berlin (2008)
42. Serrano-Cina, C.: Self organizing neural networks for financial diagnosis. *Decision Support Systems* 17(3), 227–238 (1996)
43. Thomas, J., Sycara, K.: GP and the Predictive Power of Internet Message Traffic. In: Chen, S.-H. (ed.) *Genetic Algorithms and Genetic Programming in Computational Finance*, pp. 81–102. Kluwer Academic Publishers, Dordrecht (2002)
44. Thompson, D., Thompson, S., Brabazon, A.: Predicting going concern audit qualification using neural networks. In: *Proceedings of ICAI 2007*. CSEA Press (2007)

45. Trigueros, J.: Extracting earnings information from financial statements via genetic algorithms. In: Proceedings of CIFEr 1999, pp. 281–296. IEEE Press, Los Alamitos (1999)
46. Tsang, E., Li, J.: EDDIE for Financial Forecasting. In: Chen, S.-H. (ed.) Genetic Algorithms and Genetic Programming in Computational Finance, pp. 161–174. Kluwer Academic Publishers, Dordrecht (2002)
47. Tsang, E., Martinez-Jaramillo, S.: Computational Finance. IEEE Computational Intelligence Society Newsletter, 8–13 (2004)
48. Tung, W., Quek, C.: GenSoOPATS: a brain-inspired dynamically evolving option pricing model and arbitrage system. In: Proceedings of CEC 2005, pp. 1722–1729. IEEE Press, Los Alamitos (2005)
49. Uludag, G., Uyar, A.Ş., Senel, K., Dag, H.: Comparison of evolutionary techniques for value-at-risk calculation. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 218–227. Springer, Heidelberg (2007)
50. Varetto, F.: Genetic algorithms in the analysis of insolvency risk. Journal of Banking and Finance 22(10), 1421–1439 (1998)
51. Wang, C., Zhao, X., Kang, L.: Business failure prediction using modified ants algorithm. In: Chen, S.-H., Wang, P. (eds.) Computational Intelligence in Economics and Finance. Springer, Heidelberg (2004)
52. White, A.: A genetic adaptive neural network approach to pricing options: a simulation analysis. J. of Computational Intelligence in Finance 6(2), 13–23 (1998)
53. Wong, B., Lai, V., et al.: A bibliography of neural network business applications research: 1994–1998. Computers and Operations Research 27, 1045–1076 (2000)
54. Zaiyi, G., Quek, C., Maskell, D.: FCMAC-AARS: A novel FNN architecture for stock market prediction and trading. In: Proceedings of CEC 2006, pp. 8544–8550 (2006)

Evolutionary Approaches for Estimating a Coupled Markov Chain Model for Credit Portfolio Risk Management

Ronald Hochreiter^{1,*} and David Wozabal²

¹ Department of Statistics and Decision Support Systems, University of Vienna
ronald.hochreiter@compmath.net

² Department of Statistics and Decision Support Systems, University of Vienna
david.wozabal@univie.ac.at

Abstract. The analysis and valuation of structured credit products gained significant importance during the sub-prime mortgage crisis in 2007. Financial companies still hold many products for which the risk exposure is unknown. The Coupled Markov Chain approach can be used to model rating transitions and thereby default probabilities of companies. The likelihood of the model turns out to be a non-convex function of the parameters to be estimated. Therefore heuristics are applied to find the ML estimators. In this paper, we outline the model and its likelihood function, and present a Particle Swarm Optimization algorithm, as well as an Evolutionary Optimization algorithm to maximize this likelihood function. Numerical results conclude the paper.

1 Introduction

Credit risk is one of the most important risk categories to be managed by banks. Since the seminal work of [1] a lot of research efforts have been put into the development of both sophisticated and applicable models. Furthermore, de facto standards like CreditMetrics and CreditRisk⁺ exist. Numerous textbooks provide an overview of the set of available methods, see e.g. [2], [3], and [4]. In this paper, we refer to the Coupled Markov Chain approach by [5]. The paper is organized as follows. Section 2 briefly describes the Coupled Markov Chain model and its properties, and outlines the data we used for sampling. The likelihood function, which is to be maximized is discussed in Section 3. A non-trivial method to sample from the space of feasible points for the parameters is outlined in Section 4. Two different evolutionary approaches to optimize the maximum likelihood function are presented. In Section 5 a Particle Swarm Algorithm is shown, and Section 6 introduces an Evolutionary Optimization approach. Section 7 provides numerical results for both algorithmic approaches, while Section 8 concludes the paper.

* This research was partly supported by the Austrian National Bank Jubiläumsfond Project 12306.

2 Coupled Markov Chain Model

2.1 Model Description

In the Coupled Markov Chain model proposed in [5] company defaults are modeled directly as Bernoulli events. This is in contrast to standard models used in the literature where indirect reasoning via asset prices is used to model default events of companies. The advantage of the proposed approach is that there are no heavy model assumptions necessary (normality of asset returns, no transaction costs, complete markets, continuous trading ...).

Portfolio effects in structured credit products are captured via correlations in default events. Companies are characterized by their current rating class and a second characteristic which can be freely chosen (industry sector, geographic area, ...). This classification scheme is subsequently used to model joint rating transitions of companies. We keep the basic idea of the standard Gaussian Copula model

$$X = \rho\tau + (1 - \rho)\phi,$$

where τ is the idiosyncratic part and ϕ is the systematic part determining the rating transition, while $0 \leq \rho \leq 1$ is a relative weighting factor. More specifically the Coupled Markov Chain model can be described as follows: A company n belongs to a sector $s(n)$ and is assigned to a rating class X_n^t at time t with $X_n^t \in \{0, \dots, M+1\}$ and $t : 1 \leq t \leq T$, with the credit quality decreasing with rating classes, i.e. $(M+1)$ being the default class, while 1 is the rating class corresponding to the best credit quality. The ratings of company n are modeled as Markov Chains X_n^t . The rating process of company n is determined by

- an idiosyncratic Markov Chain ξ_n^t .
- a component η_n^t which links n to other companies of the same rating class.
- Bernoulli switching variables δ_n^t which decide which of the two factors determines the rating, with $\mathbb{P}(\delta_n^{t+1} = 1) = q_{s(n), X_n^t}$, i.e. the probability of success depends on sector and rating.

All the ξ_n^t and δ_n^t are independent of everything else, while the η_n^t have a non-trivial joint distribution modeled by common Bernoulli tendency variables χ_i , $i : 1 \leq i \leq M$, such that

$$\mathbb{P}(\eta_n^t \leq X_n^t) = \mathbb{P}(\chi_{X_n^{t-1}} = 1) \text{ and } \mathbb{P}(\eta_n^t > X_n^t) = \mathbb{P}(\chi_{X_n^{t-1}} = 0),$$

i.e. the variables χ_i are indicators for a (common) non-deteriorating move of all the companies in rating class i . The rating changes of companies in different rating classes are made dependent by the non-trivial probability mass function $P_\chi : \{0, 1\}^M \rightarrow \mathbb{R}$ of the vector $\chi = (\chi_1, \dots, \chi_M)$.

The Coupled Markov Chain model is of the form:

$$X_n^t = \delta_n^t \xi_n^t + (1 - \delta_n^t) \eta_n^t.$$

and exhibits properties, which are interesting for practical application. It takes a transition matrix $P = (p_{i,j})$ as input which governs the probability of transitions for ξ_n^t and η_i^t , i.e.

$$\mathbb{P}(\xi_n^t = j) = p_{m(n),j} \text{ and } \mathbb{P}(\eta_i^t = j) = p_{i,j}.$$

The model is capable of capturing different default correlations for different sectors and rating classes, and is able to give a more accurate picture of closeness to default than the standard model by including more than two states. The overall transition probabilities of X_n again follow P , i.e.

$$\mathbb{P}(X_n = j) = p_{m(n),j}.$$

2.2 Data

Rating data from Standard & Poors has been used, whereby 10166 companies from all over the world have been considered. The data consists of yearly rating changes of these companies over a time horizon of 23 years up to the end of 2007. In total a number of 87.296 data points was used. The second characteristic is the SIC industry classification code. Sectoral information has been condensed to six categories: Mining and Construction (1), Manufacturing (2), Transportation, Technology and Utility (3), Trade (4), Finance (5), Services (6). Likewise, rating classes are merged in the following way: AAA, AA \rightarrow 1, A \rightarrow 2, BBB \rightarrow 3, BB, B \rightarrow 4, CCC, CC, C \rightarrow 5, D \rightarrow 6. These clusters allow for a more tractable model by preserving a high degree of detail. The estimated rating transition probabilities from the data are shown in Table 1.

Table 1. Estimated rating transition probabilities

$$P = \begin{pmatrix} 0.9191 & 0.0753 & 0.0044 & 0.0009 & 0.0001 & 0.0001 \\ 0.0335 & 0.8958 & 0.0657 & 0.0036 & 0.0006 & 0.0009 \\ 0.0080 & 0.0674 & 0.8554 & 0.0665 & 0.0011 & 0.0016 \\ 0.0039 & 0.0092 & 0.0794 & 0.8678 & 0.0244 & 0.0153 \\ 0.0023 & 0.0034 & 0.0045 & 0.1759 & 0.6009 & 0.2131 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

3 Maximum Likelihood Function

The approach proposed by [5] takes a Markov transition matrix $P = (p_{m_1, m_2})_{1 \leq m_1, m_2 \leq (M+1)}$ as an input, i.e.

$$\sum_{i=1}^{M+1} p_{i,m} = \sum_{i=1}^{M+1} p_{m,i} = 1, \quad \forall m : 1 \leq m \leq (M+1).$$

For $(M+1)$ rating classes, N companies and S industry sectors the parameters of the model are a matrix $Q = (q_{m,s})_{1 \leq s \leq S, 1 \leq m \leq M}$ and a probability measure P_χ on $\{0,1\}^M$ satisfying some constraints dependent on P (see problem (1)). Given rating transition data X ranging over T time periods we maximize the following monotone transformation of the likelihood function of the model

$$L(X; Q, P_\chi) = \sum_{t=2}^T \log \left(\sum_{\bar{\chi} \in \{0,1\}^M} P_\chi(\chi^t = \bar{\chi}) \prod_{s, m_1, m_2} f(x^{t-1}, s, m_1, m_2, ; Q, P_\chi) \right)$$

with

$$f(x^{t-1}, s, m_1, m_2, ; Q, P_\chi) = \begin{cases} \left(\frac{q_{m_1, s}(p_{m_1}^+ - 1) + 1}{p_{m_1}^+} \right)^{I^t}, & m_1 \geq m_2, \bar{\chi}_{m_1} = 1 \\ \left(\frac{q_{m_1, s}(p_{m_1}^- - 1) + 1}{p_{m_1}^-} \right)^{I^t}, & m_1 < m_2, \bar{\chi}_{m_1} = 0 \\ q_{m_1, s}^{I^t}, & \text{otherwise.} \end{cases}$$

where $I^t \equiv I^t(m_1, m_2, s)$ is a function dependent on the data X which takes values in \mathbb{N} , $p_m^+ = \sum_{i=1}^m p_{m,i}$ and $p_m^- = 1 - p_m^+$.

The above function is clearly non-convex and since it consists of a mix of sums and products this problem can also not be overcome by a logarithmic transform. Maximizing the above likelihood for given data X in the parameters P_χ and Q amounts to solving the following constrained optimization problem

$$\begin{aligned} & \max_{Q, P_\chi} L(X; Q, P_\chi) \\ \text{s.t. } & q_{m, s} \in [0, 1] \\ & \sum_{\bar{\chi}: \bar{\chi}_i = 1} P_\chi(\bar{\chi}) = p_{m_i}^+, \quad \forall i : 1 \leq i \leq M \\ & \sum_{\bar{\chi}: \bar{\chi}_1 = 0} P_\chi(\bar{\chi}) = 1 - p_i^+. \end{aligned} \tag{1}$$

4 Sampling Feasible Points

To sample from the space of feasible joint distributions for χ (i.e. the distributions whose marginals meet the requirements in (1)), first note that the distributions P_χ of the random variable χ are distributions on the space $\{0, 1\}^M$ and therefore can be modeled as vectors in \mathbb{R}^{2^M} . To obtain samples we proceed as follows.

1. To get a central point in the feasible region, we solve the following problem in dependence of a linear functional $\Psi : \mathbb{R}^{2^M} \rightarrow \mathbb{R}$.

$$\begin{aligned} & \max_{P_\chi} \Psi(P_\chi) \\ \text{s.t. } & q_{m, s} \in [0, 1] \\ & \sum_{\bar{\chi}: \bar{\chi}_i = 1} P_\chi(\bar{\chi}) = p_{m_i}^+, \quad \forall i : 1 \leq i \leq M \\ & \sum_{\bar{\chi}: \bar{\chi}_1 = 0} P_\chi(\bar{\chi}) = 1 - p_i^+. \end{aligned} \tag{2}$$

and call the solution set $\mathcal{S}(\Psi)$. By generating linear Ψ functionals with random coefficients and picking $x^+ \in \mathcal{S}(\Psi)$ and $x^- \in \mathcal{S}(-\Psi)$ we get vertices of the feasible set of distributions for χ modeled as a polyhedron in \mathbb{R}^{2^M} . In this way we generate a set of vertices V for the feasible region Ω of the above problem. Note that to enforce all the constraints in (2) we need $M + 1$ linear equality constraints which describe a $2^M - (M + 1)$ dimensional affine subspace in \mathbb{R}^{2^M} .

2. Get a central point in $c \in \Omega$ by defining

$$c = \frac{1}{|V|} \sum_{v \in V} v.$$

3. Sample $K \in \mathbb{N}$ directions of unit length from a spherical distribution (like the multivariate standard normal with independent components) in $\mathbb{R}^{2^M - M - 1}$ to get uniformly distributed directions. Map these directions to \mathbb{R}^{2^M} using an orthogonal basis of the affine subspace A of \mathbb{R}^{2^M} described by the last set of constraints in (2) to obtain a set of directions \mathcal{D} in A .
4. For every $d \in \mathcal{D}$ determine where the line $c + \lambda d$ meets the boundary of the feasible set of (2). Call the length of the found line segment l_d .
5. Fix a number $L \in \mathbb{N}$ and sample

$$\left\lceil \frac{l_d}{\sum_{d \in \mathcal{D}} l_d} L \right\rceil$$

points on the line l_d . In this way we get approximately KL samples for P_χ .

Contrary to obtaining samples from P_χ , getting suitable samples for Q is fairly easy, since all the components are in $[0, 1]$ and independent of each other. Note that both the set of feasible matrices Q as well as the set of feasible measures P_χ are convex sets in the respective spaces.

5 Particle Swarm Algorithm

In the following we give a brief description of the Particle Swarm Algorithm (PSA), which follows the ideas in [6].

1. Choose $\delta > 0$ and $S \in \mathbb{N}$.
2. Generate S permissible random samples $x_k = (Q^k, P_\chi^k)$ for $k = 1, \dots, S$ as described above, i.e. $q_{s,m}^k \in [0, 1]$ and P_χ is consistent with the constraints in (2). Each sample is a particle in the algorithm. Set $\hat{x}_k = x_k$ and $v_k = 0$ for all $k = 1, \dots, S$.
3. Set $\hat{g} \leftarrow \arg \min_k L(x_k)$.
4. For all particles x_k
 - (a) Let the particles fly by first computing a velocity for the k -th particle

$$v_k \leftarrow c_0 v_k + c_1 r_1 \circ (\hat{x}_k - x_k) + c_2 r_2 \circ (\hat{g} - x_k) \quad (3)$$

where c_0, c_1, c_2 are fixed constants, r_1 and r_2 are random matrices (component-wise uniform) of the appropriate dimension and \circ is the Hadamard matrix multiplication. Then a new position for the particle is found by the following assignment

$$x_k \leftarrow x_k + v_k.$$

- (b) If $L(x_k) > L(\hat{x}_k)$ then $\hat{x}_k \leftarrow x_k$.
5. $L(x_k) > L(\hat{g})$ for some x_k , then $\hat{g} \leftarrow x_k$.
6. If $\text{var}(L(x_k)) < \delta$ terminate the algorithm, otherwise go to step 3.

The main idea is that each particle k knows its best position \hat{x}_k as of yet (in terms of the likelihood function) and every particle knows the best position ever seen by any particle \hat{g} . The velocity of the particle changes in such a way that it is drawn to these positions (to a random degree). Eventually all the particles will end up close to one another and near to a (local) optimum.

Note that in step 4(a) of the above algorithm, a particle may leave the feasible region either by violating the constraints on P_χ or Q . In this case the particle *bounces off the border* and completes it's move in the modified direction. To be more precise: the particles can only leave the feasible region, by violating the constraints that either elements of Q or probabilities assigned by P_χ are no longer in $[0, 1]$ (the last two constraints in (2) can not be violated since the particles only move in the affine subspace of \mathbb{R}^{2^M} where these constraints are fulfilled).

If $x_k^i + v_k^i > 1$, determine the maximum distance λ that the particle can fly without constraint violation, i.e. set

$$\lambda = \frac{(1 - x_k^i)}{v_k^i}$$

and set $x^k \leftarrow x^k + \lambda v^k$. Now set v_k^i such that the new velocity makes the particle bounce of the constraint *as would be expected*¹ and make the rest of the move, i.e. set

$$x^k \leftarrow x^k + (1 - \lambda)v^k.$$

Obviously an implementation of the algorithm has to be able to handle multiple such bounces in one move. Since the details are straightforward, we omit them here for the sake of brevity.

6 Evolutionary Algorithm

Evolutionary algorithms are well suited to handle many financial and econometric applications, see especially [7] and [8] for a plethora of examples.

Each chromosome consists of a matrix Q and a vector P_χ . While the parameters in Q can be varied freely between 0 and 1, and the parameters P_χ do need to fulfill constraints (see above), the genetic operators involving randomness are mainly focused around the matrix Q . Therefore, four different genetic operators are used:

- **Elitist selection.** A number e of the best chromosomes are added to the new population.
- **Intermediate crossover.** c intermediate crossovers (linear interpolation) between the matrix Q_1 and Q_2 of two randomly selected parents are created

¹ In the case that the violation concerns an element of Q the modification only concerns a change of sign, i.e. $v_k^i \leftarrow -v_k^i$. If a constraint on an element of P_χ is violated, then the determination is v_k^i is a straightforward exercise in linear algebra.

using a random parameter λ between 0 and 1, i.e. two children $Q_3, P_{\chi,3}$ and $Q_4, P_{\chi,4}$ are calculated as follows:

$$Q_3 = \lambda Q_1 + (1 - \lambda)Q_2, P_{\chi,3} = P_{\chi,1},$$

$$Q_4 = (1 - \lambda)Q_1 + \lambda Q_2, P_{\chi,4} = P_{\chi,2}.$$

- **Mutation.** m new chromosomes are added by mutating a randomly selected chromosome from the parent population, and adding a factor ϕ in the range $[-0.5, 0.5]$ to the matrix Q . The values are truncated to values between 0 and 1 after the mutation.
- **Random additions.** r random chromosomes are added with a random matrix Q and a randomly selected vector P_χ from the parent population.

7 Numerical Results

Both algorithms were developed in MatLab R2007a, while the linear problems (2) were solved using MOSEK 5. A stability test has been conducted to validate the results of both optimization algorithms: the maximum (pointwise) differences of parameter estimates P_χ and Q between the different optimization runs is used to verify that these important parameters, which are e.g. used for a credit portfolio optimization procedure, do not differ significantly.

7.1 Particle Swarm Algorithm

The parameters in (3) were set to $c_0 = 0.5$, $c_1 = 1.5$ and $c_2 = 1.5$. The algorithm was made to complete 150 iterations with around 200 initial samples (where the χ are simulated on 40 lines with approximately 5 samples each). To test the stability of the algorithm 50 runs of the algorithm were performed. As can be clearly observed in Fig. 1 the likelihood of the best particle as well as the mean likelihood of the swarm converges nicely and stabilizes around iteration 25.

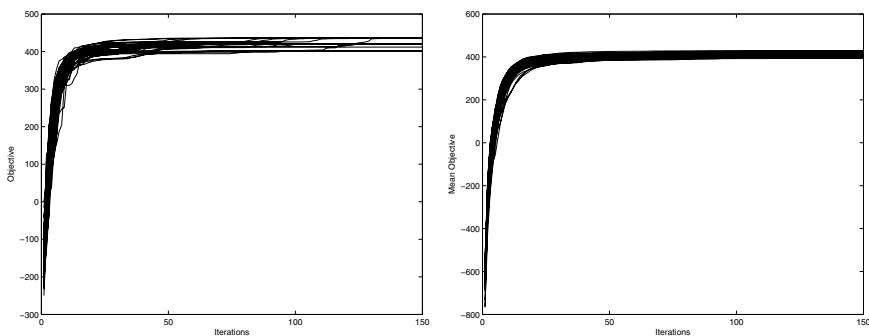


Fig. 1. Objective function: maximum per iteration (left), population mean (right)

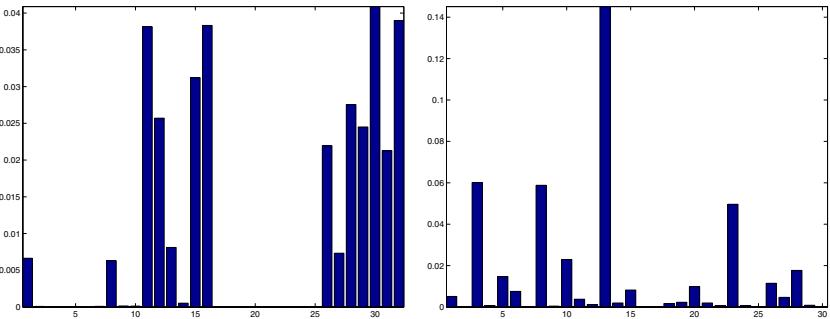


Fig. 2. Maximum (pointwise) differences of parameter estimates P_X (left) and Q (right) between the different runs for the PSA

Each run took around 1 hour to complete 150 iterations. Stability results are shown in Fig. 2.

7.2 Evolutionary Algorithm

The following parameters have been used to calculate results: The number of maximum iterations has been set to 150. Each new population consists of $e = 30$ elitist chromosomes, $c = 50$ intermediate crossovers, $m = 100$ mutations, and $r = 50$ random additions. 50 runs have been calculated, and 10 tries out of these are shown in Fig. 3 - both the maximum objective value per iteration (left) as well as the population mean (right). Due to the high number of random additions, mutations and crossovers, the mean is relatively low and does not significantly change over the iterations, which does not influence the results. The initial population size were 750 randomly sampled chromosomes, independently sampled for each try.

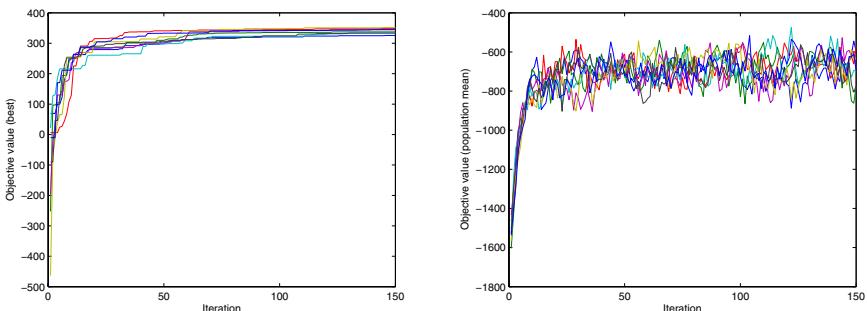


Fig. 3. Objective function: maximum per iteration (left), population mean (right)

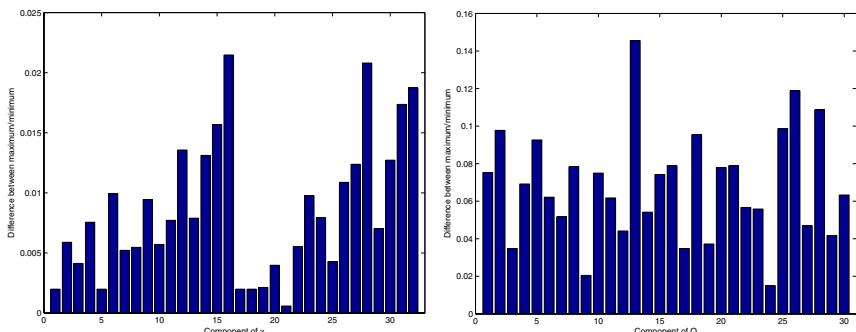


Fig. 4. Maximum (pointwise) differences of parameter estimates P_χ (left) and Q (right) between the different runs for the EA

Each run took approximately 70 minutes to complete the 150 iterations. Stability results are shown in Fig. 4.

8 Conclusion

In this paper, we presented the likelihood function for a Coupled Markov Chain model for contemporary credit portfolio risk management. We presented two different heuristic approaches for estimating the parameter of the likelihood function. Both are structurally different, i.e. the population mean of each method differs significantly. However, both are valid approaches to estimate parameters. Once the parameters are estimated, many applications are possible. One prominent example is to generate scenarios for future payment streams implied by an existing portfolio of Credit Default Swap Indices (CDX) by Monte Carlo simulation. This allows for assessing the risk of the current position and price products which might be added to the portfolio in the future and thereby determine their impact on the overall exposure.

References

1. Merton, R.: On the pricing of corporate debt: the risk structure of interest rates. *Journal of Finance* 29, 449–470 (1974)
2. Duffie, D., Singleton, K.J.: *Credit Risk: Pricing, Measurement, and Management*. Princeton University Press, Princeton (2003)
3. McNeil, A.J., Frey, R., Embrechts, P.: *Quantitative risk management*. Princeton University Press, Princeton (2005)
4. Schönbucher, P.J.: *Credit Derivatives Pricing Models: Models, Pricing, Implementation*. Wiley Finance, Chichester (2003)
5. Kaniovski, Y.M., Pflug, G.C.: Risk assessment for credit portfolios: A coupled markov chain model. *Journal of Banking and Finance* 31(8), 2303–2323 (2007)

6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE Computer Society, Los Alamitos (1995)
7. Brabazon, A., O'Neill, M. (eds.): Natural Computing in Computational Finance. Studies in Computational Intelligence, vol. 100. Springer, Heidelberg (2008)
8. Brabazon, A., O'Neill, M.: Biologically inspired algorithms for financial modelling. Natural Computing Series. Springer, Berlin (2006)

Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis

Piotr Lipinski

Institute of Computer Science,
University of Wroclaw, Wroclaw, Poland
lipinski@ii.uni.wroc.pl

Abstract. This paper discusses knowledge patterns in evolutionary learning of decision support systems for time series analysis, especially concerning time series of economical or financial data. It focuses on decision support systems, which use evolutionary algorithms to construct efficient expertises built on the basis of a set of specific expert rules analysing time series, such as artificial stock market financial experts composed of popular technical indicators analysing recent price quotations. Discovering common knowledge patterns in such artificial experts not only leads to an additional improvement of system efficiency, in particular - the efficiency of the evolutionary algorithms applied, but also reveals additional knowledge on phenomena under study. This paper shows a numer of experiments carried out on real data, discusses some examples of the knowledge patterns discovered in terms of their financial relevance as well as compares all the results with some popular benchmarks.

1 Introduction

Evolution-based models have emerged recently in the domain of time series analysis, especially time series containing economical or financial data [3], [5], [7]. There are at least three reasons for their growing popularity: robustness of prediction, possibility of dynamic market modeling, and progress in computing performances. These aspects are particularly important in many decision support systems where the market model has to be created as quickly as possible.

This paper refers to a class of decision support systems for time series analysis that bases its knowledge on evolutionarily-generated artificial experts built on a set of defined rules. These experts are generated on the basis of available past data and applied to new data until they are efficient. Since properties of analyzed data change with time, the efficiency of experts falls, at which point they should be regenerated or updated. The process of expert generation is time-consuming, prompting many ideas for its improvement. Updating an expert takes less time than regenerating one, although efficiency of an updated expert is often slightly lower than that of a newly-generated one. In this paper, we discuss methods of expert updating based on discovering and reusing frequent knowledge patterns from a set of experts generated over a specific time period.

This paper is focused on the optimization of expertise elaboration, which is discussed in the context of real-time stock trading systems and discovering frequent knowledge patterns in trading experts. The significance of delays in such systems, along with the issue of time necessary for discovering a profitable composition of indicator-based trading rules, is discussed. The proposed optimization was applied to a real-time financial data analysis system based on the genetic algorithm presented in [5].

The goal of the presented approach is to improve the throughput of the expert discovery process by discovering frequent knowledge patterns and shortening the time of expert computing by using this knowledge. The idea is to reuse previous trading experience of generated experts instead of computing experts *ex nihilo*. The trading experience can be found in patterns that are a kind of abstraction from the most recent sequence of chromosomes of the best experts. These patterns can be easily used to generate an initial population of experts that are located close to the place where the final solution might be found. Two effects of using patterns as building blocks are expected: first, reduction of the search space by decreasing the number of trading rules used for evaluation; second, acceleration of the convergence process due to the heuristics used to create an initial population of experts. The building blocks discovery process is based on sequential patterns mining algorithms applied to a list of previously generated experts [1].

This paper is structured in the following manner. The next section defines the financial knowledge base containing trading rules. Section 3 analyzes the decision making process, and Section 4 discusses an overview of evolutionary decision support systems for such a decision making process. Section 5 describes knowledge patterns and their applications in the process of expertise updating as well as validates the approach by discussing a few experiments. Finally, Section 6 concludes the paper and points out some directions for further research.

2 Financial Knowledge Base

One of popular techniques of financial time series analysis is Technical Analysis, [2], [10], which considers past stock price quotations as a principal factor affecting future price directions. It introduces many indicators, which characterize financial data, and uses them to forecast future trends and particular events like falls and rises in stock prices. Such indicators may be formalized by a concept of *a stock market trading rule*.

Definition 1. A stock market trading rule is a function

$$f : \mathcal{K} \mapsto y \in \mathbf{R} \quad (1)$$

which maps *a factual financial knowledge* \mathcal{K} to a real number y , interpreted later as a trading signal: values lower than a certain threshold α_1 correspond to a sell signal (-1.0), values greater than a certain threshold α_2 correspond to a buy signal (+1.0), and remaining values correspond to no signal (0.0) (in the

experiments, $\alpha_1 = -1$ and $\alpha_2 = 1$). In the case of Technical Analysis, *the factual financial knowledge* represents past stock price quotations. In other cases, it may also include tax rates, exchange rates, etc.

In the experiments, the financial knowledge base was composed of 250 popular trading rules, defined on the basis of [2] and [10], and the factual financial knowledge was composed of financial time series containing one-minute stock price quotations (i.e. open, high, low, close prices, volumes and index values).

3 Decision Making Process

In most cases, different trading rules produce different, often opposing, trading advices, so analysts and investors must spend some effort to transform them into one final trading decision. Such a problem may be solved by selecting an efficient set of trading rules, referred to as *a stock market trading expert*, and defining the final trading decision by the trading advice proposed by the majority of the trading rules from the chosen set.

Definition 2. Let $\mathcal{R} = \{f_1, f_2, \dots, f_d\}$ be the entire set of available trading rules. A trading expert is a subset of the entire set of trading rules

$$E \subset \mathcal{R}. \quad (2)$$

A result $E(\mathcal{K})$ of a trading expert E , for a given factual financial knowledge \mathcal{K} , is an arithmetic average of the results of the trading rules from the subset E , interpreted later as a trading signal in the same way as in the case of a single trading rule.

Consider a specific time period. For successive time instants $t_0, t_1, t_2, \dots, t_{T-1}$ of the specific time period, the trading expert E proposes trading decisions $d_0^{(E)}, d_1^{(E)}, d_2^{(E)}, \dots, d_{T-1}^{(E)}$. However, in order to apply such trading decisions on the stock market, it is necessary to define the amount of stocks to be bought or sold. Such amounts may be obtained by simulating the behavior of an hypothetical investor. This investor is given an initial endowment (c_0, s_0) with c_0 the amount of cash and s_0 the initial quantity of stocks (in the experiments, $c_0 = 10000$ and $s_0 = 100$). Since trading generates transaction costs, they are assumed proportional with rate $\tau\%$ (in simulations, $\tau = 0.2$).

At time t_0 , the investor takes a decision $d_0^{(E)}$. If the decision is to sell, i.e. $d_0^{(E)} = -1$, he sells $q\%$ of stocks, i.e. the amount of stock Δs in the investor's order is equal to

$$\Delta s = s_0 \cdot q/100. \quad (3)$$

If the decision is to buy, i.e. $d_0^{(E)} = 1$, he invests $q\%$ of money in stocks, i.e. the amount of stock Δs in the investor's order is equal to

$$\Delta s = \frac{c_0 \cdot q/100}{(1 + \tau/100) \cdot \text{Open}(t_1)}, \quad (4)$$

where $\text{Open}(t)$ denotes the opening price at date t . The parameter q in experiments equals 50, which guarantees that the investor will not empty his account too fast. The transaction is executed at time t_1 and the investor's capital changes accordingly.

Therefore, at time t_1 , the investor's capital consists of the amount of money c_1 and the amount of stocks s_1

$$c_1 = c_0 - d_0^{(E)} \cdot \Delta s \cdot \text{Open}(t_1) - \tau/100 \cdot \Delta s \cdot \text{Open}(t_1), \quad (5)$$

$$s_1 = s_0 + d_0^{(E)} \cdot \Delta s. \quad (6)$$

At time t_1 , the investor, makes a decision $d_1^{(E)}$, which is executed at time t_2 and the investor's capital again changes, and so on. Finally, c_0, c_1, \dots, c_T denote the successive cash volumes and s_0, s_1, \dots, s_T the corresponding quantities of stocks over the time period considered.

4 Evolutionary Decision Support Systems

4.1 Objectives

Although sequences of cash volumes and quantities of stocks might be used for valuation the trading expert, in practice the goal is to build a trading expert efficient over a future period with unknown stock prices, where the simulation cannot be performed. However, the future performance of an expert may be assessed on the basis of its behavior over a past period using so-called performance measures introduced by financial analysts and market traders considering not only the future return rate, but also the risk related to achieving it. There are a number of different performance measure, such as the Sharpe ratio, the Sortino ratio or the Sterling ratio [8]. In this paper, we focus on the Sharpe ratio.

Let

$$C_i = c_i + s_i \cdot \text{Open}(t_i), \quad i = 0, 1, \dots, T \quad (7)$$

$$R_i = \frac{C_i - C_{i-1}}{C_{i-1}}, \quad i = 1, 2, \dots, T \quad (8)$$

denote the investor's date- t_i wealth and the return on the portfolio for the period $[t_{i-1}; t_i]$. The Sharpe ratio for a trading expert behavior over the period $[t_0, t_0 + T]$ is defined then as

$$\varrho(E) = \frac{\mathbf{E}[R] - r_0}{\mathbf{Std}[R]}, \quad (9)$$

where $\mathbf{E}[R]$ denotes the expected return rate of the investment strategy proposed by the trading expert E over the period $[t_0, t_0 + T]$, the $\mathbf{Std}[R]$ denotes the standard deviation of the return rate and r_0 denote the reference return rate of risk-free asset.

Using the performance measure defined, discovering efficient stock market trading experts may be transformed to an optimization problem with the objective function $\varrho(E)$ over the search space of all trading experts E . Formally, let $\mathcal{R} = \{f_1, f_2, \dots, f_d\}$ be the set of all available trading rules and $\mathcal{E}(\mathcal{R})$ be the set of all available trading experts built on these rules. The objective is to find a trading expert $E \in \mathcal{E}(\mathcal{R})$ such as

$$\varrho(\tilde{E}) \leq \varrho(E) \quad (10)$$

for all $\tilde{E} \in \mathcal{E}(\mathcal{R})$, for a given training period $[t_0, t_0 + T]$ and for given parameters of the simulation described in the previous section.

4.2 Representation of Trading Experts

Each trading expert E may be represented in a natural way by a binary sequence $\mathbf{e} = (e_1, e_2, \dots, e_d)$, in such a way that, for $i = 1, 2, \dots, d$, e_i corresponds to the i -th trading rule $f_i \in \mathcal{R}$; $e_i = 0$ denotes the absence and $e_i = 1$ denotes the presence of the trading rule f_i in the set E .

Then, a one-to-one map is defined between trading experts and binary sequences of length d . Consequently, in the optimization problem defined in the previous section, the search space is simply $\{0, 1\}^d$ and the objective function is the performance $\varrho(\mathbf{e})$ of the trading expert E corresponding to the binary vector \mathbf{e} .

4.3 Discovering Trading Experts Using Evolutionary Algorithms

Building an efficient trading expert is an optimization problem with the objective function defined by the performance measure ϱ , being the Sharpe ratio, and the search space $\{0, 1\}^d$. Naturally, the objective function is defined in the context of a given set \mathcal{R} of trading rules, a given stock and a given training period. Figure 1 shows the framework of the evolutionary algorithm, based on the Simple Genetic Algorithm, [4], [9], designed to optimize the objective function ϱ with a population \mathcal{P} composed of N individuals, where M , θ_C , θ_M are some algorithm parameters.

First, the algorithm creates an initial population $\mathcal{P} = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N\} \subset \{0, 1\}^d$ at random, in such a way that each gene e_{ij} of each chromosome \mathbf{e}_i is generated randomly using the uniform distribution, i.e. the probability of $e_{ij} = 0$ and the probability of $e_{ij} = 1$ are both equal to 0.5. After creation, the population is evaluated.

Afterwards, the population evolves under the influence of four evolutionary operators, namely *parent selection*, *crossover*, *mutation* and *replacement*, until a termination condition is satisfied. It terminates when it completes a specific number of iterations or when there is no increase in objective function values over a specific number of recent iterations. Due to size constraints, complete specifications of evolutionary operators are omitted and may be found in [4] or [9].

```

SIMPLE-GENETIC-ALGORITHM( $\varrho, N, M, \theta_C, \theta_M$ )
1  $\mathcal{P} \leftarrow \text{RANDOM-POPULATION}(N);$ 
2  $\text{POPULATION-EVALUATION}(\mathcal{P}, \varrho);$ 
3 while not  $\text{TERMINATION-CONDITION}(\mathcal{P})$ 
4 do
5    $\mathcal{P}^{(P)} \leftarrow \text{PARENT-SELECTION}(\mathcal{P}, M);$ 
6    $\mathcal{P}^{(C)} \leftarrow \text{CROSSOVER}(\mathcal{P}^{(P)}, \theta_C);$ 
7    $\text{MUTATION}(\mathcal{P}^{(C)}, \theta_M);$ 
8    $\text{REPLACEMENT}(\mathcal{P}, \mathcal{P}^{(C)});$ 
9    $\text{POPULATION-EVALUATION}(\mathcal{P}, \varrho);$ 

```

Fig. 1. The Simple Genetic Algorithm designed to optimize the objective function ϱ with a population \mathcal{P} composed of N individuals, where M, θ_C, θ_M are some algorithm parameters

5 Frequent Knowledge Patterns

In practice, trading experts, built by the evolutionary algorithm on the basis of financial information related to a specific time period, become out-of-date and lose profitability when the stock market differs significantly from the initial state. As the stock market changes incessantly, trading experts must be built anew very often.

However, studies on binary representations of trading experts suggest that such new trading experts may be built with reusing some knowledge patterns extracted from previous trading experts. In terms of evolutionary algorithms, for a trading expert being a sequence of binary digits, a knowledge pattern is a subsequence of such a sequence. A frequent knowledge pattern is a subsequence which occurs in many previous trading experts, in other words, it is a common combination of specific trading rules. Reusing frequent knowledge patterns would lead to *a priori* excluding or including such trading rules in the trading expert and running the evolutionary algorithm only to check the remaining part of trading rules. It would result in a reduction of the search space and an acceleration of computations.

In order to discover frequent knowledge patterns, a number of experiments were carried out on 4 financial time series containing one-minute stock price quotations from the Paris Stock Exchange (namely, AXA, Peugeot, Renault and STMicroelectronics). In all the experiments, the same set of 250 trading rules was used and the evolutionary algorithm was run with the same configuration (the population size was 50, the iteration number was 400). Although each stock was considered separately, trading experts and knowledge patterns were studied independently, similar results were obtained. Due to size constraints, the further discussion focuses only on the case of Peugeot.

First, for each minute of a chosen time period of 6 hours, a trading expert was built independently without any reusing of knowledge patterns. Each trading expert was built on the basis of financial information consisting of stock quotations from preceding 4 hours.

Figure 2 shows frequencies of usage of 250 trading rules in 360 trading experts built for successive minutes of the chosen time period. It is easy to see that there is a number of frequently chosen trading rules (e.g. trading rules included in more than 300 trading experts) as well as a number of infrequently chosen trading rules (e.g. trading rules included in less than 60 trading experts). Such trading rules may constitute the 360-minute knowledge pattern.

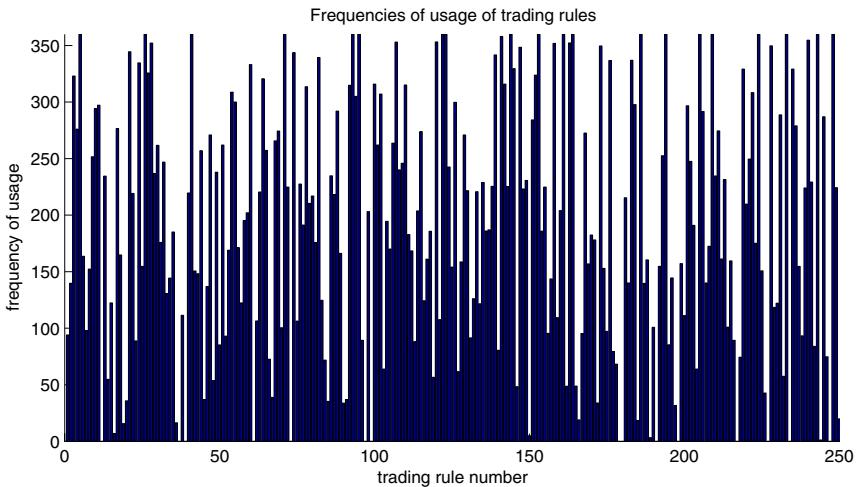


Fig. 2. Frequencies of usage of 250 trading rules in 360 trading experts

Unfortunately, only 10% to 15% of trading rules were either constantly used, or not used at all, in trading experts built during time periods of 6 hours. In order to find more knowledge patterns, shorter time periods were also considered.

Figure 3 shows an extraction of usage of trading rules in trading experts built for successive minutes of a shorter time period of 15 minutes. In order to illustrate the concept of a knowledge pattern, the figure shows only 18 trading rules. It is easy to see that some trading rules (namely, 1, 2, 4, 16, 17) were either constantly used, or not used at all, in trading experts and may form the knowledge pattern.

In the experiments, 15-minute knowledge patterns include approximately 40% of trading rules. For comparison, 10-minute knowledge patterns and 5-minute knowledge patterns cover 60% and 85% of trading rules, respectively.

For the same experiment, Table 1 shows a full summary of frequencies of usage of trading rules. It is easy to see that there are $71 + 67 = 138$ trading rules, which are either constantly used, or not used at all, in trading experts.

Based on the observations, the two last intervals, 10 and 5 minutes, were considered to be too short to produce a reasonable knowledge pattern. In addition, in these two cases the search space is too restricted, inhibiting the exploration

Time	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	
15:45	1	0	1	1	1	0	0	1	0	0	0	1	0	1	0	1	1	0	
15:46	1	0	1	1	1	0	0	1	0	0	0	1	0	1	0	1	1	0	
15:47	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	1	1	0	
15:48	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	
15:49	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	
15:50	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	
15:51	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	
15:52	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	
15:53	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	
15:54	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
15:55	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
15:56	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
15:57	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
15:58	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
15:59	1	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0
Freq. (N)	15	0	9	15	3	1	1	3	12	6	1	8	5	13	5	15	15	0	

Fig. 3. An extraction of usage of 18 trading rules in trading experts built during a time period of 15 minutes

Table 1. The frequency of the usage of 250 trading rules over a period of 15 minutes

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
number of trading rules occurring N times in the last 15 trading experts	67	10	6	10	4	6	12	10	9	13	6	3	7	10	6	71

by evolutionary operators. The 15-minute knowledge pattern seems to be a compromise among the quality of trading, the computing time, and the space exploration facility. In this case, according to numerous trials, the time of trading expert generation was shortened by approximately 50%.

In the experiment, the 15-minute pattern was computed each minute and then used to generate a new trading expert. The search space was reduced. Nevertheless, a large number of variant experts were evolved using only the part of the chromosome not covered by the pattern. To avoid premature convergence and local solutions, after every 10 minutes for a period of 5 minutes the experts were generated according to the classical algorithm. The process of pattern discovery was then run again. This solution makes the expert design extensible, reusable and adaptable to specific market situations.

The results of these experiments are encouraging. Many tests show that the pattern captures a subset of an already successful group of rules applied to a recurring trading problem that arises within a certain context. On average, decreasing the time of expert generation has considerably improved the overall throughput of the system. Two aspects of this improvement should be underlined. First, the system is now able to generate a trading expert in approximately 12 seconds after the latest market observation. Consequently, the interval for quote aggregation may be reduced from one minute to less than 15 seconds.

Table 2. Excess of the return rate of the investment strategies defined by trading experts over the return rate of the B&H strategy

extremely positive B&H, i.e. $0.01 \leq B\&H$	$0.0323\% \pm 0.0051$	$0.0412\% \pm 0.0013$
positive B&H, i.e. $0.00 \leq B\&H < 0.01$	$0.0225\% \pm 0.0033$	$0.0268\% \pm 0.0024$
negative B&H, i.e. $-0.01 \leq B\&H < 0.00$	$0.0265\% \pm 0.0075$	$0.0283\% \pm 0.0082$
extremely negative B&H, i.e. $B\&H < -0.01$	$0.0142\% \pm 0.0029$	$0.0157\% \pm 0.0015$

Second, using the one-minute deferred quotations, a trading agent may obtain advice for more securities traded on the market.

Finally, in order to study the financial aspect of the approach proposed, investment strategies defined by trading experts built with reusing knowledge patterns were compared with original investment strategies defined by trading experts built anew without any reusing of knowledge patterns in terms of profitability. 40 experiments were carried out, for different stocks and different time periods of 30 minutes. In each experiment, for each minut of the time period, two trading experts were built – one with reusing knowledge patterns and one without it.

Table 2 reports the excess of the return rate of the investment strategies over the return rate of the simple Buy-and-Hold (B&H) strategy which consists in investing all the capital in stocks at the start of the time period and keeping it until the end of the time period. Results usually depend on stock market conditions, so experiments were divided into 4 groups according to the state of the stock market, defined by ranges of B&H values over the time period. It is easy to see that the performance of the approach proposed depends on stock market conditions. However, in most cases, it seems not to be worse than the original investment strategy and usually outperforms the B&H strategy.

6 Conclusions

This paper concerns discovering frequent knowledge patterns from artificial experts in evolutionary decision support systems for time series analysis and reusing these patterns for improving the evolutionary process of building efficient experts.

At the moment, patterns are looked for based on a set of genetic experts that come from the last 15 minutes. It is tempting to shorten this period. This would, however, cause the pattern to become larger, but this may provoke premature convergence of the genetic algorithm, and hence the mutation operators would have to be modified in order to avoid focusing on a small part of the search space.

On the other hand, one can analyze patterns occurring over longer periods in order to promote or eliminate rules that are permanently chosen or rejected. In this case, it would appear to be necessary to develop a strategy which would allow a rule to come back to the previous status after a certain time. Otherwise, a rule might be eliminated forever, even if it would turn out to be efficient in the future.

Reduction of the computing time enables the further development of the decision support system. Currently, the data aggregation period is equal to one minute, which implies that the system generates advice every minute. Since the decision making process has been accelerated, it is tempting to lessen the aggregation period so that the system will react faster, for instance, every 30 seconds.

References

1. Agrawal, R.A., Srikant, R.: Mining Sequential Patterns. In: Proc. of the International Conference on Data Engineering, ICDE, Taipei, Taiwan (1995)
2. Colby, W., Meyers, T.: The Encyclopedia of Technical Market Indicators, Down Jones-Irwin (1990)
3. Dempster, M.A.H., Jones, C.M.: A Real-Time Adaptive Trading System using Genetic Programming. Quantitative Finance 1, 397–413 (2001)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
5. Korczak, J., Lipinski, P.: Evolutionary Building of Stock Trading Experts in a Real-Time System. In: Proceedings of the 2004 Congress on Evolutionary Computation, CEC 2004, pp. 940–947. IEEE, Los Alamitos (2004)
6. Lipinski, P.: Discovering Stock Market Trading Rules using Multi-Layer Perceptrons. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 1114–1121. Springer, Heidelberg (2007)
7. Lipinski, P.: ECGA vs. BOA in Discovering Stock Market Trading Experts. In: Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2007, pp. 531–538. ACM, New York (2007)
8. Lipinski, P., Korczak, J.: Performance Measures in an Evolutionary Stock Trading Expert System. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3039, pp. 835–842. Springer, Heidelberg (2004)
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York (1994)
10. Murphy, J.: Technical Analysis of the Financial Markets, NUIF (1998)

Predicting Turning Points in Financial Markets with Fuzzy-Evolutionary and Neuro-Evolutionary Modeling

Antonia Azzini, Célia da Costa Pereira, and Andrea G.B. Tettamanzi

Università degli Studi di Milano
Dipartimento di Tecnologie dell'Informazione
via Bramante 65, I-26013 Crema, Italy
{azzini,pereira,tettamanzi}@dti.unimi.it

Abstract. Two independent evolutionary modeling methods, based on fuzzy logic and neural networks respectively, are applied to predicting trend reversals in financial time series, and their performances are compared. Both methods are found to give essentially the same results, indicating that trend reversals are partially predictable.

1 Introduction

Even the most casual observer of a financial time series will notice that prices of financial instruments move up and down [8]; furthermore, this behavior happens and can be observed at all scales [9]. However, price movements are not regular and look unpredictable.

In general, market action consists of alternating up-trends and down-trends, separated by turning points, which correspond to maxima and minima. A trader able to buy at minima and sell at maxima, i.e., trade exactly at the turning points, would gain the maximum profit possible. For this reason, the main objective of financial market forecasting techniques is to call turning points consistently and correctly. Two approaches for calling turning points in a financial time series are possible:

- reveal turning points when they occur, or just after they have occurred — this is what most technical analysis indicators are all about, starting from simple moving averages to the most sophisticated indicators;
- predict the price at which the next turning point will most likely occur — this is the approach we follow in this paper.

To do that, we summarize the past history of the series up to the last confirmed turning point by applying a noise-eliminating filter. The output of the filter is given as input to a predictive model, whose output provides an estimate of the price at which the next turning point is going to happen.

Biologically inspired methods have become extremely popular as tools for modeling financial markets [5,6]; these include evolutionary algorithms, neural

networks, and fuzzy logic. In this paper we employ two types of models, namely fuzzy rule bases, i.e., sets of fuzzy IF-THEN rules, and feedforward neural networks. Both types of models are designed and optimized by means of evolutionary algorithms. The performance of models of the two types are compared with the aim of assessing which one is more suited to the task.

The paper is organized as follows: Section 2 states the problem, Section 3 provides a brief description of the two modeling methods, and Section 4 reports the experiments and discusses their results. Section 5 concludes.

2 Problem Description

To eliminate noise, the time series of prices of the financial instrument under study is pre-processed by applying the *zig-zag* filter [1] with a threshold θ . The *zig-zag* decomposes the input time series into a series of alternating up- and down-swings. The length of each swing is given by the price difference of its ends. To make this datum adimensional, the length of each swing is divided by the length of the previous swing. The series $\{r_t\}_t$ of the resulting ratios is the output of the noise-eliminating filter.

The problem of predicting the price at which the next turning point will occur can thus be transformed into the problem of predicting the next ratio of the series of ratios of swing lengths, since the price of a turning point x_{t+1} can be calculated by knowing the price of the two previous turning points x_{t-1} and x_t and the ratio $r_t = \frac{|x_{t+1} - x_t|}{|x_t - x_{t-1}|}$.

The working hypothesis is that such prediction can be done at any time by considering the n most recent ratios. In other words, we are searching the space of all autoregressive models of $\{r_t\}_t$ of order n .

A baseline for any model is provided by the simplest model possible, namely a model that always predicts $E[r_t]$, without taking the last n ratios into account. $E[r_t]$ can be estimated by taking the longest available time series for the financial instrument considered and computing the average ratio of a swing length to the length of the previous swing. Such a model has a mean absolute error $E[|r_t - E[r_t]|]$ and a mean square error $E[(r_t - E[r_t])^2] = \text{Var}[r_t]$, while the relative mean absolute error corresponds to $E[|r_t - E[r_t]|]/E[r_t]$. This, by the way, is the best performance one would expect from a predictive model if the time series under observation were completely random or, which is roughly equivalent [9], if the market were perfectly efficient.

3 Evolutionary Optimization

3.1 Fuzzy Rule Base Optimization

Data mining is a process aimed at discovering meaningful correlations, patterns, and trends between large amounts of data collected in a dataset. A model is determined by observing past behavior of a financial instrument and extracting

the relevant variables and correlations between the data and the dependent variable. We describe below a data-mining approach based on the use of EAs, which recognize patterns within a dataset, by learning models represented by sets of fuzzy rules.

Fuzzy Models. A model is described through a set of fuzzy rules, made by one or more antecedent clauses (“IF . . .”) and a consequent clause (“THEN . . .”). Clauses are represented by a pair of indices referring respectively to a variable and to one of its fuzzy sub-domains, i.e., a membership function.

Using fuzzy rules makes it possible to get homogenous predictions for different clusters without imposing a traditional partition based on crisp thresholds, that often do not fit the data, particularly in financial applications. Fuzzy decision rules are useful in approximating non-linear functions because they have a good interpolative power and are intuitive and easily intelligible at the same time. Their characteristics allow the model to give an effective representation of the reality and simultaneously avoid the “black-box” effect of, e.g., neural networks.

The intelligibility of the model is useful for a trader, because understanding the rules helps the user to judge if a model can be trusted.

The Evolutionary Algorithm. The described approach incorporates an EA for the design and optimization of fuzzy rule-based systems originally developed to learn fuzzy controllers [10,11], then adapted for data mining, which has already been used for financial modeling by two of the authors [7].

A model is a rule base, whose rules comprise up to four antecedent and one consequent clause each. Input and output variables are partitioned into up to 16 distinct linguistic values each, described by as many membership functions. Membership functions for input variables are trapezoidal, while membership functions for the output variable are triangular. Models are encoded in three main blocks:

1. a set of trapezoidal membership functions for each input variable; a trapezoid is represented by four fixed-point numbers;
2. a set of symmetric triangular membership functions, represented as an area-center pair, for the output variable;
3. a set of rules, where a rule is represented as a list of up to four antecedent clauses (the IF part) and one consequent clause (the THEN part); a clause is represented by a pair of indices, referring, respectively, to a variable and to one of its membership functions.

An island-based distributed EA is used to evolve models. The sequential algorithm executed on every island is a standard generational replacement, elitist EA. Crossover and mutation are never applied to the best individual in the population.

The recombination operator is designed to preserve the syntactic legality of models. A new model is obtained by combining the pieces of two parent models. Each rule of the offspring model can be inherited from one of the parent models with probability 1/2. When inherited, a rule takes with it to the offspring model

all the referred domains with their membership functions. Other domains can be inherited from the parents, even if they are not used in the rule set of the child model, to increase the size of the offspring so that their size is roughly the average of its parents' sizes.

Like recombination, mutation produces only legal models, by applying small changes to the various syntactic parts of a fuzzy rulebase.

Migration is responsible for the diffusion of genetic material between populations residing on different islands. At each generation, with a small probability (the migration rate), a copy of the best individual of an island is sent to all connected islands and as many of the worst individuals as the number of connected islands are replaced with an equal number of immigrants.

A detailed description of the algorithm and of its genetic operators can be found in [10].

3.2 Neuro Genetic Optimization

The second evolutionary approach [2,3,4] considered in this work evolves a population of NNs, encoding multilayer perceptrons (MLPs), a type of feed-forward NN. The algorithm is based on the joint optimization of structure and weights, and uses the error back-propagation (BP) algorithm to decode a *genotype* into a *phenotype* NN. Accordingly, it is the genotype which undergoes the genetic operators and which reproduces itself, whereas the phenotype is used *only* for calculating the genotype's fitness. The rationale for this choice is that the alternative of using BP, applied to the genotype, as a kind of "intelligent" mutation operator would boost exploitation while impairing exploration, thus making the algorithm too prone to being trapped in local optima. The individuals are not constrained to a pre-established topology, and the population is initialized with different hidden layer sizes and different numbers of neurons for each individual according to two exponential distributions, in order to maintain diversity among all of them in the new population. Such dimensions are not bounded in advance, even though the fitness function may penalize large networks. The number of neurons in each hidden layer is constrained to be greater than or equal to the number of network outputs, in order to avoid hourglass structures, whose performance tends to be poor. Indeed, a layer with fewer neurons than the outputs destroys information which later cannot be recovered.

The evolutionary process adopts the convention that a lower fitness means a better NN, mapping the objective function into an error minimization problem. Therefore the fitness is proportionate to the mean square error and to the cost of the considered network.

Evolutionary Process. In the overall evolutionary process the population is randomly created, initialized, and the genetic operators are then applied to each network until termination conditions are not satisfied.

At each generation, a population consisting of the best $\lfloor n/2 \rfloor$ individuals is selected by truncation from a population of size n ; the remaining NNs are then duplicated in order to replace those eliminated, and finally, the population is

randomly permuted. Elitism allows the survival of the best individual unchanged into the next generation and the solutions to get better over time. Then, for all individuals of the population the algorithm mutates the weights and the topology of the offsprings, trains the resulting network, calculates fitness on the test set, and finally saves the best individual and statistics about the entire evolutionary process.

Weights mutation perturbs the weights of the neurons before performing any structural mutation and applying BP to train the network. All the weights and the corresponding biases are updated by using variance matrices and evolutionary strategies applied to the synapses of each NN, in order to allow a control parameter, like mutation variance, to self-adapt rather than changing their values by some deterministic algorithm. Finally, the topology mutation is implemented with four types of mutation by considering neurons and layer addition and elimination. The addition and the elimination of a layer and the insertion of a neuron are applied with three independent probabilities, while the elimination of a neuron is carried out only if the contribution of that neuron is negligible with respect to the overall network output.

4 Experiments and Results

We have considered the time series of daily prices of the S&P500 index from September 26, 1985 to October 31, 2008, filtered with two different values of θ , namely $\theta = 0.01$ and $\theta = 0.02$. For $\theta = 0.01$, $E[r_t] = 1.2918$, $\text{Var}[r_t] = 1.1062$, relative error 0.5586; for $\theta = 0.02$, $E[r_t] = 1.2418$, $\text{Var}[r_t] = 0.6895$, relative error 0.49624. With $n = 12$, the application of the filter produces datasets of, respectively, 2,769 and 1,095 records.

Following the commonly accepted practice of machine learning, the problem data are partitioned into training, test and validation sets, used, respectively for training, to stop learning avoiding overfitting, and to test the generalization capabilities of each model.

We have set aside, for validation, the 200 most recent records for $\theta = 0.01$ and the 100 most recent records for $\theta = 0.02$. Of the remaining records, a random 10% is used as test set by the fuzzy-evolutionary method and 18% and 10% respectively by the neuro-genetic method.

We have performed 20 runs for either dataset with the neuro-genetic method and 10 runs for either dataset with the fuzzy-evolutionary method, which is more demanding in terms of computational resources. The results of those runs are shown in Table 1, that also shows a consistent, if not dramatic, improvement with respect to the baseline model, whose prediction is always $E[r_t]$, for both methods and both datasets.

The fact that two independent methods, both based on global optimization methods like evolutionary algorithms, yet using two radically different “languages” to represent models, have obtained almost identical results, can be taken

Table 1. A comparison of the results obtained by the two methods on the two datasets generated for $\theta = 0.01$ and $\theta = 0.02$ when applied to the validation set (out of sample)

Method	Dataset →			$\theta = 0.01$			$\theta = 0.02$		
	mean	stdev	best	mean	stdev	best	mean	stdev	best
Fuzzy-Evolutionary	0.4720	0.0055	0.4657	0.4110	0.0049	0.4057			
Neuro-Evolutionary	0.4813	0.0096	0.4740	0.40307	0.0250	0.3958			

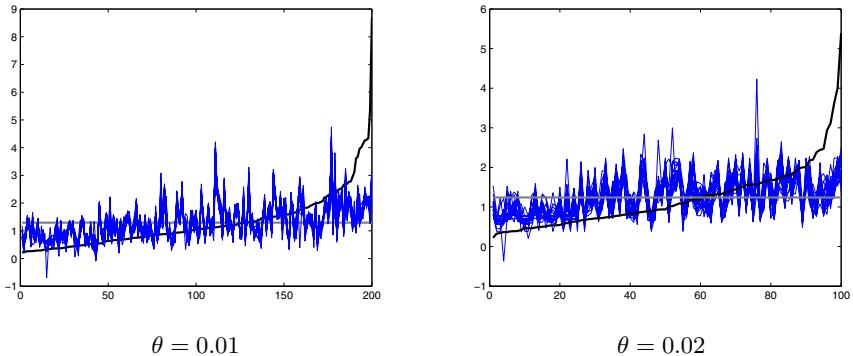


Fig. 1. Performance on the validation set (out-of-sample data) of the neural networks evolved by independent runs of the neuro-genetic method for the two datasets considered. The records of the validation set have been sorted by increasing r_t . The thick black line is the actual r_t , the thick light-grey line is $E[r_t]$, while the other lines represent the predictions of each neural network.

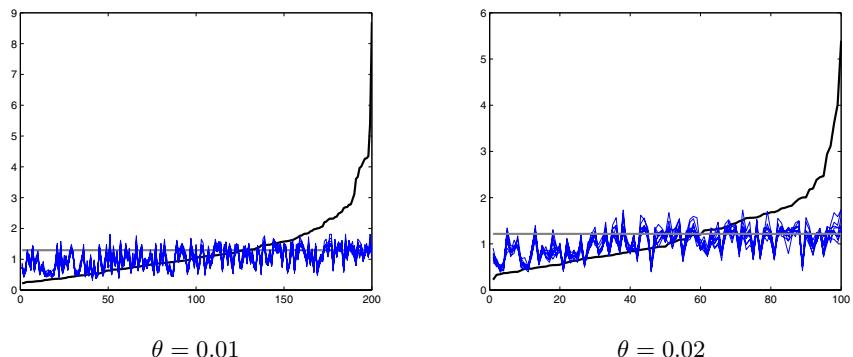


Fig. 2. Performance on the validation set (out-of-sample data) of the fuzzy rule bases evolved by independent runs of the fuzzy-evolutionary method for the two datasets considered. The records of the validation set have been sorted by increasing r_t . The thick black line is the actual r_t , the thick light-grey line is $E[r_t]$, while the other lines represent the predictions of each fuzzy model.

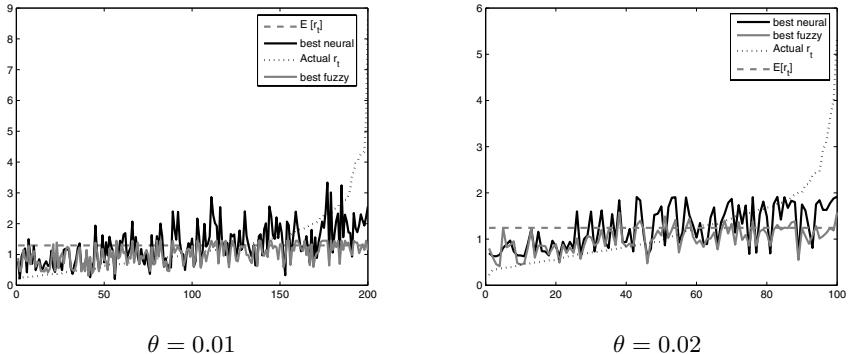


Fig. 3. Comparison of the best results on the validation set of the fuzzy- and the neuro-evolutionary approaches. The dashed black line is the actual r_t , the dashed grey line is $E[r_t]$, while the grey line represents the best fuzzy model and the black line the best neural model.

as a guarantee that there is very little room for further improvement, if any, and their results are very close to the optimum.

Figures 1 and 2 plot, respectively, the performance of the best neural networks and fuzzy rule-based models produced by each run, while Figure 3 compares the two best solutions obtained from the fuzzy and neuro-genetic approaches over the validation set for, respectively, $\theta = 0.01$ and $\theta = 0.02$.

The best topologies obtained from the neuro-genetic approach for each of the two datasets are shown in Figures 4 and 5, corresponding, respectively, to $\theta = 0.01$ and 0.02 . An interesting aspect that can be highlighted is that, while the first corresponds to a more usual network with normal connections, in the second network (see Figure 5) the last hidden connection is set, together with the bias of the last node, to a negative value. Such connection could represent a reversal (through a NOT operator) of the information flow; however, it seems to be an effect of the backpropagation used to train the networks.

The best fuzzy rule bases obtained by the fuzzy-evolutionary approach for either dataset are shown in Figures 6 and 7. It can be observed that both rule bases feature a sort of *default* rule, which always fires, and sets the prediction of $\text{swing}(t)$ to 1.44 for $\theta = 0.01$ and 1.57 for $\theta = 0.02$, while all the remaining rules appear to be there to recognize and handle significant patterns where different prediction can be made. Interestingly, the rule base in Figure 6 uses but the three most recent swings to predict the next one; the most recent swings have a prevailing role in the rule base in Figure 7 as well, although “older” swings are looked at in a few rules.

Although at first sight the predictions provided by the models found by both methods may appear disappointing, a closer examination of the graphs in Figures 1 and 2 reveals something interesting: a clear tendency can be observed for models evolved by independent runs of both methods to provide similar

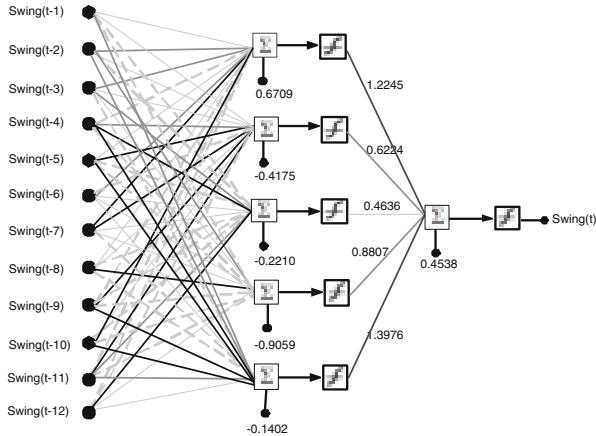


Fig. 4. Topology of the best neural network for $\theta = 0.01$. The thick black line refers to connection weights $w \geq 1$. The thick dark grey line corresponds to values defined as $0.5 \leq w < 1$, while the light grey line to those defined as $0 \leq w < 0.5$. Finally the dashed light grey line corresponds to negative correlations $w < 0$.

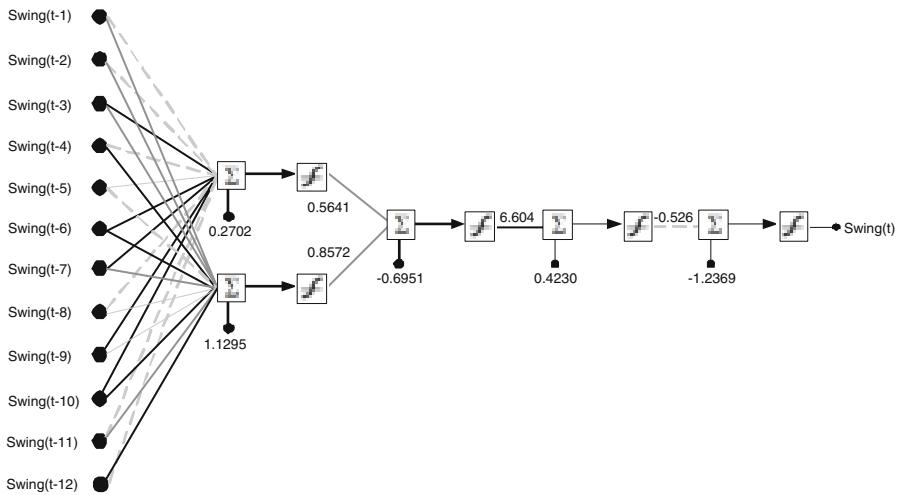


Fig. 5. Topology of the best neural network for $\theta = 0.02$. As reported in Figure 4, the thick black line refers to connection weights $w \geq 1$. The thick dark grey line corresponds to $0.5 \leq w < 1$, and the light grey line to $0 \leq w < 0.5$. Finally the dashed light grey line corresponds to negative correlations $w < 0$.

predictions for the same records. This cannot be a coincidence, but it must be understood as evidence that the models are striking trade-offs among errors committed when predicting r_t in context that look similar as far as the recent previous history of the time series is concerned.

```

IF TRUE THEN swing( $t$ ) is 1.44
IF swing( $t - 1$ ) is medium-large THEN swing( $t$ ) is 0.12
IF swing( $t - 3$ ) is very-large AND swing( $t - 11$ ) is small-to-medium THEN swing( $t$ ) is 0.12
IF swing( $t - 1$ ) is medium AND swing( $t - 2$ ) is medium-to-large THEN swing( $t$ ) is 0.12
IF swing( $t - 1$ ) is medium-to-huge AND swing( $t - 2$ ) is medium THEN swing( $t$ ) is 0.12
IF swing( $t - 1$ ) is medium-to-huge THEN swing( $t$ ) is 0.12
IF swing( $t - 1$ ) is medium AND swing( $t - 2$ ) is medium-to-large THEN swing( $t$ ) is 0.12

```

Fig. 6. The best fuzzy rule base for $\theta = 0.01$. The linguistic values have been manually given meaningful names to improve readability.

```

IF swing( $t - 7$ ) is large AND swing( $t - 11$ ) is medium-large AND swing( $t - 5$ ) is large
AND swing( $t - 2$ ) is medium THEN swing( $t$ ) is 8.65
IF swing( $t - 7$ ) is large AND swing( $t - 5$ ) is medium-large AND swing( $t - 1$ ) is large
THEN swing( $t$ ) is 15
IF TRUE THEN swing( $t$ ) is 1.57
IF swing( $t - 1$ ) is large AND swing( $t - 8$ ) is medium-small THEN swing( $t$ ) is 0.13
IF swing( $t - 1$ ) is small-to-large AND swing( $t - 2$ ) is medium THEN swing( $t$ ) is 0.13
IF swing( $t - 2$ ) is medium THEN swing( $t$ ) is 0.13
IF swing( $t - 6$ ) is around 11 AND swing( $t - 12$ ) is between 6 and 10
AND swing( $t - 8$ ) is medium-small THEN swing( $t$ ) is 0.13
IF swing( $t - 1$ ) is small-to-large THEN swing( $t$ ) is 0.13
IF swing( $t - 1$ ) is large AND swing( $t - 9$ ) is small THEN swing( $t$ ) is 0.13
IF swing( $t - 1$ ) is large THEN swing( $t$ ) is 0.13

```

Fig. 7. The best fuzzy rule base for $\theta = 0.02$. The linguistic values have been manually given meaningful names to improve readability.

5 Conclusion and Future Work

Two independent evolutionary modeling methods have been applied to predicting trend reversals in financial time series.

The results obtained indicate that the lengths of price swings of financial instruments (detailed results have been shown regarding the S&P500 index, but the time series of all the other instruments we have tried to apply the same approach to show the same behavior) follow a largely, but not exclusively, random pattern. However, the non-random part of their behavior can be modeled and predicted. Whether this predictable part of a financial time series behavior can be effectively exploited for gaining excess returns is an open question, but we believe our results are yet another small piece of evidence against the efficient market hypothesis.

References

1. Achelis, S.B.: Technical analysis from A to Z. Probus Publisher, Chicago (1995)
2. Azzini, A., Tettamanzi, A.: A neural evolutionary approach to financial modeling. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2006, vol. 2, pp. 1605–1612. Morgan Kaufmann, San Francisco (2006)

3. Azzini, A., Tettamanzi, A.: Neuro-genetic single position day trading. In: Workshop Italiano di Vita Artificiale e Computazione Evolutiva, WIVACE 2007 (2007)
4. Azzini, A., Tettamanzi, A.: Evolutionary Single-Position Automated Trading. In: Proceedings of European Workshop on Evolutionary Computation in Finance and Economics, EVOFIN 2008, pp. 1605–1612 (2008)
5. Brabazon, A., O'Neill, M.: Biologically Inspired Algorithms for Financial Modelling. Springer, Berlin (2006)
6. Brabazon, A., O'Neill, M.: Natural Computing in Computational Finance. Springer, Berlin (2008)
7. da Costa Pereira, C., Tettamanzi, A.: Fuzzy-evolutionary modeling for single-position day trading. In: Brabazon, A., O'Neill, M. (eds.) Natural Computing in Computational Finance. Studies in Computational Intelligence, vol. 100, pp. 131–159. Springer, Berlin (2008)
8. Herbst, A.: Analyzing and Forecasting Futures Prices. Wiley, New York (1992)
9. Peters, E.: Chaos and Order in the Capital Markets, 2nd edn. Wiley, New York (1996)
10. Tettamanzi, A., Poluzzi, R., Rizzotto, G.G.: An evolutionary algorithm for fuzzy controller synthesis and optimization based on SGS-Thomson's W.A.R.P. fuzzy processor. In: Zadeh, L.A., Sanchez, E., Shibata, T. (eds.) Genetic algorithms and fuzzy logic systems: Soft computing perspectives. World Scientific, Singapore (1996)
11. Tettamanzi, A.: An evolutionary algorithm for fuzzy controller synthesis and optimization. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 5(5), pp. 4021–4026. IEEE Systems, Man, and Cybernetics Society (1995)
12. Yao, X.: Evolutionary Optimization. Kluwer Academic Publishers, Norwell (2002)
13. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks 8(3), 694–713 (1997)

Comparison of Multi-agent Co-operative Co-evolutionary and Evolutionary Algorithms for Multi-objective Portfolio Optimization

Rafał Dreżewski, Krystian Obrocki, and Leszek Siwik

Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
drezew@agh.edu.pl

Abstract. Co-evolutionary techniques makes it possible to apply evolutionary algorithms in the cases when it is not possible to formulate explicit fitness function. In the case of social and economic simulations such techniques provide us tools for modeling interactions between social or economic agents—especially when agent-based models of co-evolution are used. In this paper agent-based versions of multi-objective co-operative co-evolutionary algorithms are applied to portfolio optimization problem. The agent-based algorithms are compared with classical versions of SPEA2 and NSGA2 multi-objective evolutionary algorithms.

1 Introduction

Evolutionary algorithms are heuristic techniques which can be used for finding approximate solutions of global optimization problems. Evolutionary algorithms were also applied with great success to multi-modal and multi-objective problems (for example compare [2]), however in these cases some special mechanisms should be used in order to obtain good results. These are of course mechanisms specific for problems being solved but it seems that very important mechanisms in the case of multi-modal and multi-objective problems are the ones that maintain population diversity, because we are interested in finding not the single solution (as in the case of global optimization problems) but rather the whole sets of solutions.

Co-evolution is one of the mechanisms that can support maintaining of population diversity (see [10]). Another effect of applying co-evolutionary mechanisms is that we do not have to explicitly formulate the fitness function—we can just encode solutions in the genotypes and approximate fitness values for individuals on the basis of tournaments (competitive co-evolutionary algorithms) or co-operation (co-operative co-evolutionary algorithms).

Agent-based evolutionary algorithms are decentralized models of co-evolutionary computations. In fact two approaches are possible when we try to mix agent-based and evolutionary paradigms. In the first one agents are used to “manage” the evolutionary computations. In such approach each agent has the population of individuals inside of it, and this sub-population is evolved with the use of a standard evolutionary algorithm. Agents themselves can migrate within the computational environment, from one computational node to another, trying to utilize free computational resources.

The example of the second approach is *co-evolutionary multi-agent system (Co-EMAS)* which results from the realization of (co-)evolutionary processes in multi-agent system (for example see [4]). In such systems agents “live” within the environment. All agents possess the ability to reproduce, they can compete for limited resources present within the environment, and die when they run out of resources. Multi-agent co-evolutionary algorithms based on CoEMAS model (utilizing different co-evolutionary mechanisms like predator-prey, host-parasite, co-operation, etc.) were already applied to multi-objective problems (for example see [7]) and to generating investment strategies ([6]).

Described above approaches can be mixed. For example, one can imagine the system in which agents serve as management layer, and individuals, which “live” within such agents are also agents. They can also migrate from one management agent to another and make independently all decisions.

Agent-based co-evolutionary systems have some distinguishing features, among which the most interesting seem to be: the possibility of constructing hybrid systems, in which many different computational intelligence techniques are used together within one coherent agent-based computational model, relaxation of computational constraints (because of decentralization of evolutionary computations), and the possibility of introducing new evolutionary operators and social relations, which were hard or impossible to introduce in the case of “classical” evolutionary computations.

The system presented in this paper uses agents for managing evolutionary computations. Additionally, agent-based co-operative co-evolutionary approach is adapted for solving the multi-objective problem of portfolio optimization. The results of experiments are used to compare proposed agent-based co-operative co-evolutionary algorithm, agent-based co-operative versions of well known SPEA2 and NSGA2 algorithms, and original versions of SPEA2 and NSGA2.

2 Agent-Based Co-operative Co-evolutionary System

The general architecture of the system presented in this paper was described with details in [5]. Here we rather concentrate on the details of algorithms used in the system. The system was implemented with the use of Java based framework jAgE ([1]). This framework is particularly suitable for implementing agent-based evolutionary algorithms because it provides all necessary elements like environment composed of computational nodes, agents, basic mechanisms for agent-agent and agent-environment interactions.

The system which we describe here has five implemented algorithms. Agent-based algorithms utilize agent layer for managing evolutionary computations. Three versions of agent-based co-evolutionary algorithms were implemented: co-operative co-evolutionary multi-agent algorithm (*CCEA-jAgE*), agent-based co-operative co-evolutionary version of NSGA2 algorithm (*CCNSGA2-jAgE*), and agent-based co-operative co-evolutionary version of SPEA2 algorithm (*CCSPEA2-jAgE*). Also two classical multi-objective evolutionary algorithms were implemented: NSGA2 and SPEA2 (details of these algorithms may be found in [2]).

Algorithm 1. The first step of the aggregate agent

```

1 for  $a \leftarrow a_1$  to  $a_n$  do /* $a_i$  is the  $i$ -th computational agent*/
2   | receive the initial population  $P_a^0$  from agent  $a$ ; /* $P_a^0$  is the sub-population of
    | agent  $a$  in step 0*/
3 end
4  $C =$  aggregation of the solutions from  $P^0$ ; /* $C$  is the set of complete
   | solutions (co-operations) consisted of the individuals coming from
   | different sub-populations*/
5 calculate the contribution of each of the individuals in the co-operation;
6 for  $a \leftarrow a_1$  to  $a_n$  do
7   | send the sub-population  $P_a^0$  to agent  $a$ ;
8 end
9  $A^0 =$  choose the non-dominated solutions from  $C$ ; /* $A$  is the set of
   | non-dominated solutions found so far*/

```

Algorithm 2. Step of the computational agent

```

1 receive sub-population  $P^t$  from aggregate agent; /* $P^t$  is the sub-population in
   | time  $t$ */
2 compute the fitness of individuals from  $P^t$  on the basis of their contribution to the whole
   | solution quality;
3  $P^{t+1} \leftarrow \emptyset$ ;
4 while  $P^{t+1}$  is not full do
5   | select parents from  $P^t$ ;
6   | generate offspring from parents and apply recombination;
7   |  $P^{t+1} = P^{t+1} +$  offspring;
8 end
9 mutate individuals from  $P^{t+1}$ ;
10 send  $P^{t+1}$  to aggregate agent;

```

In the **co-operative co-evolutionary multi-agent algorithm (CCEA-jAgE)**, which is based on the co-operative algorithm proposed in [9], there are computational agents which have individuals inside of them. Computational agents are located within the computational nodes of the jAgE platform—these nodes can be located on the same machine or on different machines connected with network. Agent-aggregate (which is the kind of a central point of the system) is responsible for the creation of complete solutions and maintaining the set of non-dominated solutions found so far.

The first step of aggregate agent is shown in alg. 1. After each iteration of the algorithm computational agents send their sub-populations to the aggregate agent for evaluation (the single step of computational agent is shown in alg. 2). Aggregate agent then chooses individuals that would form the complete solutions (see alg. 3). Next, the aggregate agent calculates contributions of each individual from each sub-population to the whole solution of the problem (alg. 4). This is done by calculating the values of all criteria for all solutions. Then aggregate sends back the subpopulations to their respective computational agents in order to compute the fitness values of individuals. Such

Algorithm 3. Step of the aggregate agent managing the computations

```

1 while stop condition is not fulfilled do
2   for  $a \leftarrow a_1$  to  $a_n$  do
3     | receive sub-population  $P_a^t$  from agent  $a$ ;
4   end
5   for  $a \leftarrow a_1$  to  $a_n$  do
6     |  $P_a^{t+1}$  = select individuals for new generation from  $P_a^{t-1} \cup P_a^t$ ;
7   end
8    $C^{t+1} \leftarrow$  complete solutions formed from  $P^{t+1}$ ;
9   calculate the contribution of individuals coming from different species to the whole
    solution quality;
10  for  $a \leftarrow a_1$  to  $a_n$  do
11    | send the sub-population  $P_a^{t+1}$  to the agent  $a$ ;
12  end
13  update the set of non-dominated solutions  $A^{t+1}$  with the use of  $C^{t+1}$ ;
14 end

```

an approach results in minimizing the communication overhead generated by agents sending communicates via network. Also, in such a way the aggregate agent is able to update the set of non-dominated solutions found so far—it inserts into the set all non-dominated solutions from the current population and then removes from this set all solutions dominated by the newly inserted ones.

CCNSGA2-jAgE—agent-based co-operative co-evolutionary version of NSGA2 algorithm—is possible to obtain via the proper configuration of the previously described algorithm. The fitness computation in agent-based co-evolutionary NSGA2 is realized with the use of non-dominated sorting and crowding distance metric (see [2]). The aggregate agent joins the populations of parents and offspring, and chooses (on the basis of elitist selection and within each sub-population separately) individuals which will form the next generation sub-population. This sub-population will be then used for the creation of co-operations (complete solutions).

In the case of **agent-based co-operative co-evolutionary version of SPEA2 algorithm (CCSPEA2-jAgE)** some modifications of the algorithms presented previously had to be introduced. SPEA2 uses additional external set of solutions during the process of evaluating individuals ([11]), so in the agent-based co-evolutionary version of SPEA2 algorithm each computational agent has its own, local, external set of solutions (IA) used during the fitness estimation. This set is also sent to the aggregate agent. The step of selecting individuals to the next generation sub-population may be now omitted by aggregate agent, because IA^t is the set of parents—and it is obtained from computational agent. In order to create the set of complete solutions C^t and compute contributions of the individuals to the quality of the complete solutions, the aggregates are created from the individuals coming from populations P^t and IA^t .

Computational agent updates the archive IA^{t+1} with the use of mechanisms from SPEA2 ([11]). Parents are selected from IA^{t+1} and children generated with the use of recombination operator are inserted into P^{t+1} (offspring population). Then mutation is applied to the individuals from set P^{t+1} and it is sent to aggregate agent together with the individuals from IA^{t+1} .

Algorithm 4. Calculating the contribution of individuals coming from different species to the whole solution quality

```

1 for species  $P_s \leftarrow P_0$  to  $P_n$  do
2   | choose representatives  $r_s$  from  $P_s$ ;
3 end
4  $C \leftarrow \emptyset$ ;
5 for species  $P_s \leftarrow P_0$  to  $P_n$  do
6   | for individual  $i_s \leftarrow i_0$  to  $i_N$  do
7     |   |  $c_{pool} \leftarrow \emptyset$ ;
8     |   | for  $j \leftarrow 1$  to  $|r_s|$  do
9       |     |   |  $x \leftarrow$  aggregation of  $i_s$  with the representatives of the other species;
10      |     |   | compute  $F(x)$ ;
11      |     |   |  $c_{pool} \leftarrow c_{pool} + \{x\}$ ;
12    |   | end
13    |   |  $x \leftarrow$  solution chosen from  $c_{pool}$ ;
14    |   |  $C \leftarrow C + \{x\}$ ;
15    |   |  $F(x)$  is set as the contribution of individual  $i_s$  to the whole solution quality;
16  | end
17 end
18 return  $C$ 

```

3 The Experiments

The algorithms presented in the previous section were preliminary assessed with the use of commonly used multi-objective test DTLZ functions ([3]). The results of these experiments are presented in [5]. Generally speaking, the results obtained with the use of agent-based algorithms (especially CCEA-jAgE) were comparable, and in the case of some problems better, than those obtained with the use of SPEA2 and NSGA2. In this section we will present the results of experiments with the problem of multi-objective portfolio optimization.

In all compared algorithms (CCEA, CCNSGA2, CCSPEA2, NSGA2 and SPEA2) the binary representation was used. One point crossover and bit inversion was used as genetic operators. As the selection mechanism tournament selection with elitism was used. The size of the population was set to 50. In order to minimize the differences between algorithms the values of crucial (and specific to each algorithm) parameters were obtained during preliminary experiments.

The results presented in this section include Pareto frontiers generated by the algorithms. Also, in order to better compare the generated results, hypervolume metric was used. Hypervolume (HV) metric ([2]) allows to estimate both the convergence to the true Pareto frontier as well as distribution of solutions over the whole approximation of the Pareto frontier. Hypervolume describes the area covered by solutions of obtained approximation of the Pareto frontier result set. For each found non-dominated solution, hypercube is evaluated with respect to the fixed reference point.

In the case of optimizing investing portfolio complete solution is represented as a p -dimensional vector. Each decision variable represents the percentage participation of

Algorithm 5. The algorithm (based on the one-factor Sharpe model) of computing the expected risk level and income expectation

- 1 Compute the arithmetic means on the basis of rate of returns;
- 2 Compute the value of α coefficient $\alpha_i = \overline{R_i} - \beta_i \overline{R_m}$;
- 3 Compute the value of β coefficient $\beta_i = \frac{\sum_{t=1}^n (R_{it} - \overline{R_i})(R_{mt} - \overline{R_m})}{\sum_{t=1}^n (R_{mt} - \overline{R_m})^2}$;
- 4 Compute the expected rate of return of asset i $R_i = \alpha_i + \beta_i R_m + e_i$;
- 5 Compute the variance of random index $s_{e_i}^2 = \frac{\sum_{t=1}^n (R_{it} - \alpha_i - \beta_i R_m)^2}{n-1}$;
- 6 Compute the variance of market index $s_m^2 = \frac{\sum_{t=1}^n (R_{mt} - \overline{R_m})^2}{n-1}$;
- 7 Compute the risk level of the investing portfolio $\beta_p = \sum_{i=1}^p (\omega_i \beta_i)$;
- 8 $s_{e_p}^2 = \sum_{i=1}^p (\omega_i^2 s_{e_i}^2)$;
- 9 $risk = \beta_p^2 s_m^2 + s_{e_p}^2$;
- 10 Compute the portfolio rate of return $R_p = \sum_{i=1}^p (\omega_i R_i)$;

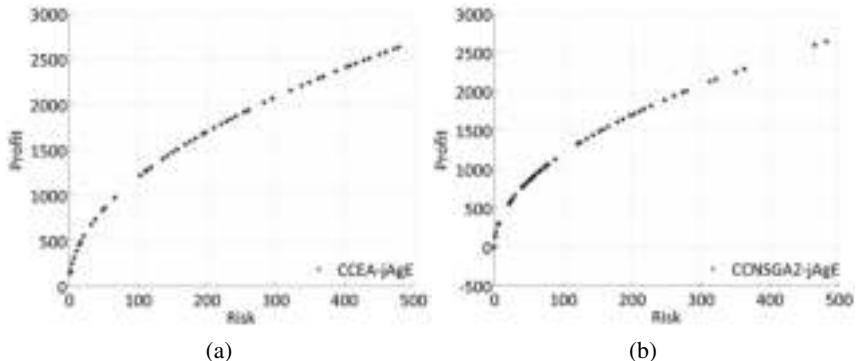


Fig. 1. Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations for CCEA (a) and CCNSGA2 (b)

i -th ($i \in 1 \dots p$) share in the whole portfolio. The problem is described with details in [8] (in this paper the agent-based predator-prey algorithm was used to solve this problem). Below we will present only the most important issues.

During presented experiments Warsaw Stock Exchange quotations from 2003-01-01 until 2005-12-31 were taken into consideration. Simultaneously, the portfolio consists of the three or seventeen stocks quoted on the Warsaw Stock Exchange. As the market index WIG20 has been taken into consideration.

During experiments one-factor Sharpe model was used. This model was also used in [8] (in this work also comparison to other models and explanation why this particular model was used during experiments may be found). The algorithm (based on the one-factor Sharpe model) of computing the expected risk level and income expectation related to the portfolio of p assets is presented in alg. 5. The meaning of symbols used in the alg. 5, are as follows: p is the number of assets in the portfolio, n is the number of periods taken into consideration (the number of rates of return taken to the model), α_i, β_i are coefficients of the equations, ω_i is the percentage participation of i -th asset in

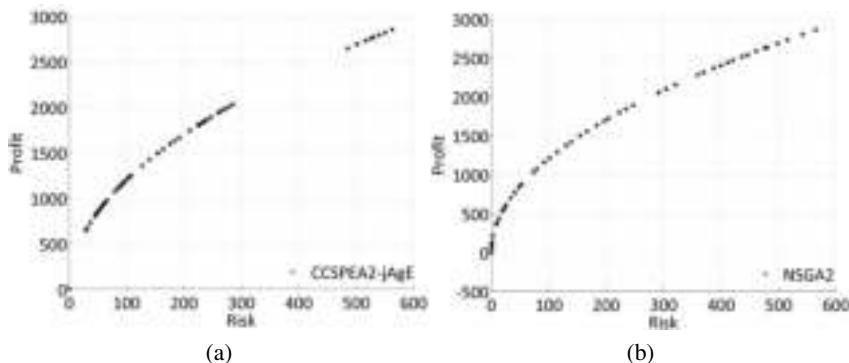


Fig. 2. Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations for CCSPEA2 (a) and NSGA2 (b)

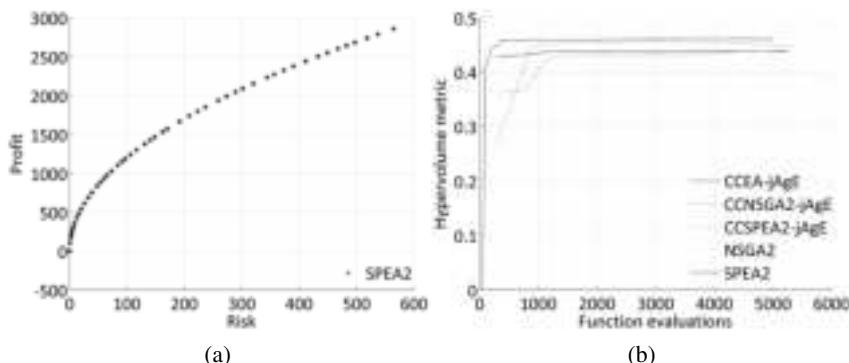


Fig. 3. Pareto frontier obtained for 3 stocks problem after 5000 fitness function evaluations for SPEA2 (a) and the values of HV metrics (b)

the portfolio, e_i is random component of the equation, R_{it} is the rate of return in the period t , R_{mt} is the rate of return of market index in period t , R_m is the rate of return of market index, R_i is the rate of return of the i -th asset, R_p is the rate of return of the portfolio, s_i^2 is the variance of the i -th asset, $s_{e_i}^2$ is the variance of random index of the i -th asset, $s_{e_p}^2$ is the variance of the portfolio, \bar{R}_i is arithmetic mean of rate of return of the i -th asset, \bar{R}_m is arithmetic mean of rate of return of market index.

The goal of the optimization is to maximize the portfolio rate of return and minimize the portfolio risk level. The task consists in determining values of decision variables $\omega_1 \dots \omega_p$ forming the vector $\Omega = [\omega_1, \dots, \omega_p]^T$, where $0\% \leq \omega_i \leq 100\%$ and $\sum_{i=1}^p \omega_i = 100\%$ and $i = 1 \dots p$ and which is the subject of minimization with respect of two criteria $F = [R_p(\Omega) * (-1), \text{risk}(\Omega)]^T$.

The Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations in typical experiment are presented in figures 1, 2, and 3a. The figure 3b shows the average values of HV metric from 15 experiments for all compared algorithms. In this case (3 stocks) results are quite comparable for all implemented algorithms. Slightly

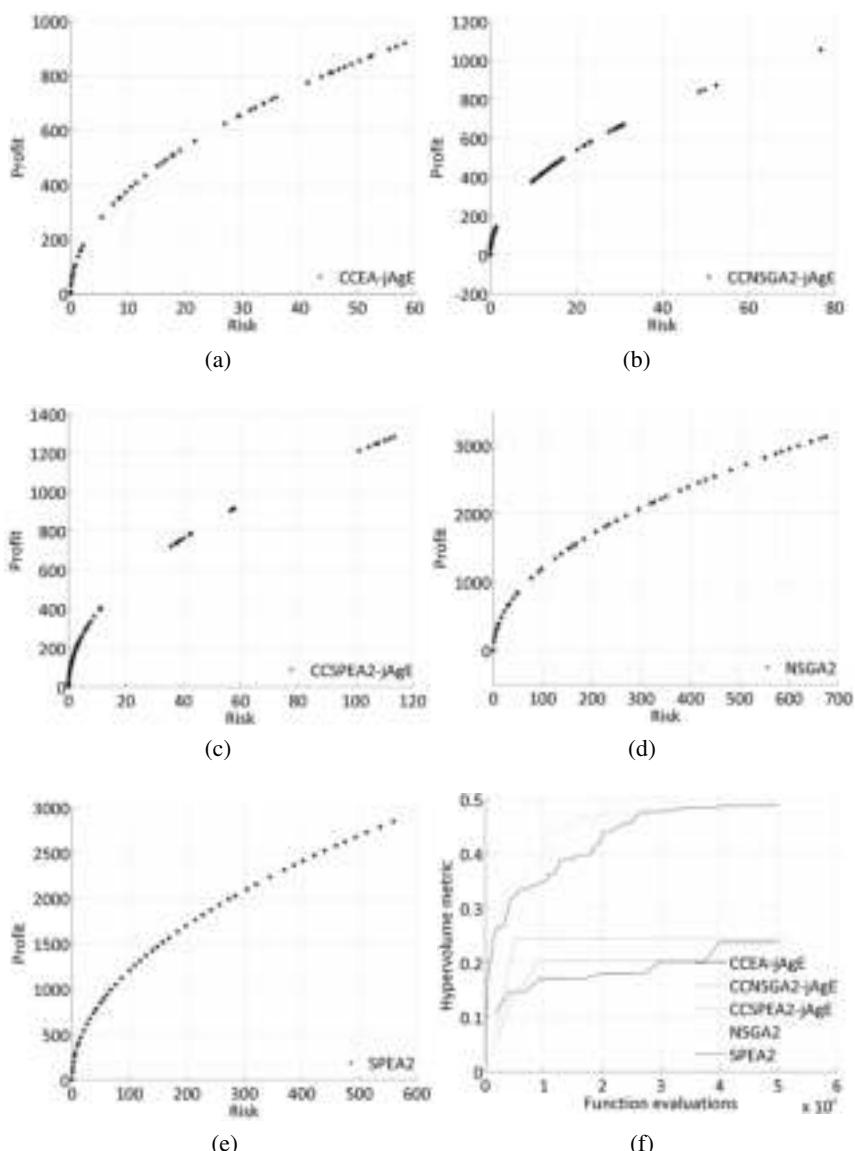


Fig. 4. Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations for CCEA (a), CCNSGA2 (b), CCSPEA2 (c), NSGA2 (d), SPEA2 (e) and the values of HV metrics (f)

worse results were obtained with the use of agent-based versions of SPEA2 and NSGA2 algorithms.

The Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations in typical experiment are presented in figures 4a–4e. In the figure 4f the average

values of HV metric are presented (these are also average values from 15 experiments). In the case of the problem with 17 stocks the best results were obtained with the use of NSGA2 and SPEA2. When we look at the presented sample Pareto frontiers CCEA-jAgE algorithm formed quite comparable frontier—but the average value of HV metric was worse than in the case of NSGA2 and SPEA2. Agent-based versions of SPEA2 and NSGA2 decisively obtained worse results than other algorithms.

4 Summary and Conclusions

In this paper we have presented agent-based co-operative co-evolutionary algorithm for solving multi-objective problems. Also four other algorithms were implemented within the agent-based system: NSGA2, SPEA2 and agent-based co-operative co-evolutionary versions of these two state-of-the-art algorithms. In the paper [5] these algorithms were compared with the use of standard multi-objective test problems—DTLZ functions [3]. In this paper we have applied the system to the multi-objective problem of constructing optimal portfolio.

In the case of DTZL problems the winner was CCEA-jAgE algorithm—agent-based version of co-operative co-evolutionary algorithm (see [5]). In the case of optimal portfolio problem used in this paper the results are mixed. In the case of portfolio consisted of three stocks the results were rather comparable in the case of all algorithms—only agent-based versions of SPEA2 and NSGA2 algorithms obtained slightly worse results. In the case of seventeen stocks decisive winners are SPEA2 and NSGA2—especially when the values of HV metric are taken into consideration. Presented results lead to the conclusion that certainly more research is needed in the case of multi-objective agent-based techniques. But also it can be said that the results presented here (and in [5]) show that neither classical nor agent-based techniques can alone obtain good quality results for all kinds of multi-objective problems. We must carefully choose the right technique on the basis of the problem characteristics because there are no universal solutions. The algorithm that can obtain very good solutions for all types of multi-objective problems simply does not exist and we think that results presented here and in other our papers show this fact clearly.

When the future work is taken into consideration we can say that certainly presented agent-based algorithms will be further developed and tested on other multi-objective problems. Another direction of the research is (mentioned in section 1) the other way of merging multi-agent and evolutionary paradigms—the way in which agents are not used as the management layer but as the individuals that live, evolve and co-operate or compete with each other.

References

1. Agent-based evolution platform (jAgE), <http://age.iisg.agh.edu.pl>
2. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. Technical report, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (2001)

4. Dreżewski, R.: Co-evolutionary multi-agent system with speciation and resource sharing mechanisms. *Computing and Informatics* 25(4), 305–331 (2006)
5. Dreżewski, R., Obrocki, K.: Co-operative co-evolutionary approach to multi-objective optimization. In: ICANNGA 2009 (submitted, 2009)
6. Dreżewski, R., Sepielak, J.: Evolutionary system for generating investment strategies. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008*. LNCS, vol. 4974, pp. 83–92. Springer, Heidelberg (2008)
7. Dreżewski, R., Siwik, L.: Agent-based co-operative co-evolutionary algorithm for multi-objective optimization. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2008*. LNCS, vol. 5097, pp. 388–397. Springer, Heidelberg (2008)
8. Dreżewski, R., Siwik, L.: Co-evolutionary multi-agent system for portfolio optimization. In: Brabazon, A., O'Neill, M. (eds.) *Natural Computation in Computational Finance*, pp. 271–299. Springer, Heidelberg (2008)
9. Keerativuttumrong, N., Chaiyaratana, N., Varavithya, V.: Multi-objective co-operative co-evolutionary genetic algorithm. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) *PPSN 2002*. LNCS, vol. 2439, pp. 288–297. Springer, Heidelberg (2002)
10. Paredis, J.: Coevolutionary algorithms. In: Bäck, T., Fogel, D., Michalewicz, Z. (eds.) *Handbook of Evolutionary Computation*, 1st supplement. IOP Publishing and Oxford University Press (1998)
11. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report TIK-Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (2001)

Dynamic High Frequency Trading: A Neuro-Evolutionary Approach

Robert Bradley, Anthony Brabazon, and Michael O'Neill

Natural Computing Research and Applications Group
University College Dublin, Ireland

`robert.bradley@ucdconnect.ie, anthony.brabazon@ucd.ie, m.oneill@ucd.ie`

Abstract. Neuro-evolution of augmenting topologies (NEAT) is a recently developed neuro-evolutionary algorithm. This study uses NEAT to evolve dynamic trading agents for the German Bond Futures Market. High frequency data for three German Bond Futures is used to train and test the agents. Four fitness functions are tested and their out of sample performance is presented. The results suggest the methodology can outperform a random agent. However, while some structure was found in the data, the agents fail to yield positive returns when realistic transaction costs are included. A number of avenues of future work are indicated.

1 Introduction

Bond markets play a significant role in capital allocation in developed economies. As an illustration of the scale of these markets, the total value of outstanding marketable bond debt in the US was approximately \$29.2 trillion [1] as of the 30th of September 2007. In comparison, the total amount of marketable equity of all US companies on the same date was approximately \$22.3 trillion [1]. Multiple issuers use bond markets to raise funding including companies, financial institutions, government agencies and central government. Given the scale and liquidity of bond markets they attract substantial trading interest.

The construction of an intraday trading system for bond futures is a difficult task as the time series of prices from this market is quite volatile and we do not have strong theory to describe the exact nature of the price-generation process. This suggests that a model-induction methodology such as a multi-layer perceptron may have utility in uncovering this process from the underlying data. A practical difficulty in developing a multi-layer perceptron is that the creation of a quality network for a specific application can be time-consuming. Hence, there has been significant interest in the possibility of automating some or all of this development process by means of evolving neural net structures. A key issue in doing this efficiently is the matching of the design of the diversity-generating operator(s) to the choice of representation for the neural network. This is not a trivial task and a diverse range of approaches have been suggested. In this study we adopt the NEAT methodology [14,13] which adopts a principled approach to this issue.

Evolutionary algorithms have been used to evolve trading rules for a number of markets with varying degrees of success. Previous studies which apply evolutionary algorithms to the problem of trading try a number of fitness functions which vary from a simple total return value, to a risk adjusted return based ratio. This study employs a neuro-evolutionary methodology to model a dateset of high frequency bond futures data in an attempt to automatically trade the market. In addition, we investigate whether or not better out of sample performance can be gained by using different fitness functions. Four such functions were tested and their performance is compared against a zero intelligence random agent.

1.1 Structure of Paper

The rest of this paper is organised as follows. The next section provides an overview of the bond futures market along with a short literature review of EC applied to trading. We then outline our experimental approach. This is followed by a results section. The paper finishes with a conclusion and future work section.

2 Background

2.1 Trading Bond Futures

A futures contract is a standardised agreement between two counter-parties where the buyer (seller) is agreeing to take delivery of (deliver) a specific quantity of the *underlying* (for example, a financial security, a foreign currency or a commodity etc.) on a specific date in the future (called the *maturity date*), for a price agreed upon now. Futures can, for example, be used by producers and consumers in order to hedge their respective future income and exposures by providing assurance as to the future price they will receive/pay for some item. This is essential for businesses when trying to manage projected cashflows and associated risks. Traders in financial markets can also use futures to hedge certain exposures in their portfolio. For example, a Fixed Income dealer might want to protect a portfolio of government bonds from adverse changes in interest rates by purchasing, or selling, bond futures. Futures can also be used by speculators who want to express a view on the direction of the market.

In the case of fixed income futures the underlying is a fixed income product such as a government bond. The future's market price is quoted in the same way as the underlying bond, i.e. as a percentage of the face value of the bond.

In this study we look at three particular fixed income futures which derive their value from German government bonds and which are traded on Eurex; the *Euro-Schatz* (FGBS), *Euro-Bobl* (FGBM), and *Euro-Bund* (FGBL) Bond Futures. All three futures trade the next 4 maturities in a quarterly (March, June, September, December) delivery cycle and expire on the 10th day of the delivery month.

2.2 Literature Review

There have been numerous previous studies which employ EC methodologies such as GA and GP to evolve trading rules. Most of these works have either

Table 1. An overview of the fitness functions employed by previous work

Reference	Market	Frequency	Fitness Function
Dittmar & Neely et al 97 [8]	FX	Daily	Log Returns
Allen & Karjalainen 99 [9]	Stock Index	Daily	Log Returns
Neely 99 [11]	Stock Index	Daily	Log Returns & Sharpe Ratio & X*
Dempster & Jones 00 [6]	FX	Intraday	Sterling Ratio
Dempster et al 01 [7]	FX	Intraday	Log Returns
Bhattacharya & Pictet et al 02 [3]	FX	Intraday	Sharpe Ratio & UtilityFunc
Brabazon & O'Neill 04 [4]	FX	Daily	Return - MaxDrawdown
Neely & Weller 03 [10]	FX	Intraday	Log Returns
Dempsey & O'Neill et al 04 [5]	Stock Index	Daily	Sharpe Ratio
Azzini & Tettamanzi 08 [2]	Stocks	Daily	Sortino Ratio

evolved trading rules for spot foreign exchange or stock markets. There has been little or no investigation into applying these algorithms to the problem of trading derivatives such as futures.

A key decision when setting up an EC experiment is choosing a suitable fitness function. This function acts as a selection criterion which biases the stochastic search process. A number of different fitness functions have been utilised in previous work. These can be broadly categorised as being risk, or non risk adjusted measures of fit.

Table 1 lists some of the more recent applications of EC to trading. Most of the literature referenced in table 1 does not compare the out of sample performance under different fitness functions, with the exception of [3] where it is found that risk adjusted metrics yield more stable out of sample performance. This study aims to investigate whether or not different fitness functions do in fact yield different out of sample behavior. In addition, we are applying our chosen methodology to a dataset of high frequency Bond Futures data to see if profit can be captured from the futures market.

3 Experimental Approach

3.1 Methodology

In this study we employ the neuro-evolution of augmenting topologies (NEAT) methodology to evolve multi-layer perceptrons (MLP) for the purposes of developing a trading system for the German bond market. NEAT was developed in 2002 by Stanley and Miikkulainen [14,13] and attempts to overcome the problems of evolving MLPs using a direct encoding of an MLP structure. NEAT simultaneously evolves both MLP topology and weights. Proponents of NEAT claim that it:

1. applies a principled method of crossover (i.e. attempts to overcome the permutation problem),
2. protects structural innovations (i.e. attempts to overcome noisy fitness problem), and
3. grows MLPs from a minimal structure (i.e. attempts to ensure that final MLP is no more structurally complex than necessary).

In order to achieve this NEAT uses three mechanisms, a novel MLP encoding which allows for ‘sensible crossover’, a speciation concept which offers some protection for structural innovations, and a seeding process whereby all initial MLP structures are of minimal size.

3.2 Data Review

The dataset used in this study consists of 5,000 5 minute bars of intraday data for the three German bond futures mentioned in Sect. 2; the Euro-Schatz, Euro-Bobl, and Euro-Bund. A bar contains a value for the open, high, low, and close prices for the interval, and also the number of contracts traded in the 5 minute period (volume). The series exhibit a variety of price behaviors, including bullish, bearish, and choppy periods. This varied behavior poses a difficult learning environment for NEAT.

The bond futures market is one of the most active fixed income markets. Table 2 shows a number of basic statistics on the volume of activity (volume of contracts traded) on the market for the period 13/06/2008 to 25/07/2008. The Bund tends to be more volatile, and more heavily traded than the Schatz and Bobl. Total volume for the given period of trading in the Schatz is over 15 million contracts, compared to the Bund where over 25 million contracts were traded. The average number of contracts traded per 5 minute block ranges from 2,887 (Bobl) to 4,812 (Bund). Each of these corresponds to a bond with a face value of Euro 100,000 and hence represents a substantial ‘gross’ position in the underlying (bond) instrument.

Table 2. Volume statistics (measured in number of contracts traded per 5 min block)

	Schatz	Bobl	Bund
Total	15,186,286	14,432,952	24,057,734
Mean	3,037	2,887	4,812
Std	3,817	3,146	5,367
Max	59,738	40,240	57,818
Min	1	1	1

A move in a bond future price from 99.31 to 99.32 corresponds to a full ‘tick’. As already noted, the Bund moves in full ticks, but the Schatz and Bobl contracts move in half ticks. Table 3 describes the behavior of the first differences of the closing price data. The volatility varies across contracts, with the Bund being the most volatile with an average move of 0.274 of a tick between each 5 minute block. As can be seen, the standard deviation of the number of tick changes between five minute blocks is quite high relative to the mean, illustrating the volatile nature of price changes in even short time periods for these futures.

Table 3. Ticks in five minute block

	Schatz	Bobl	Bund
Mean	0.0099	0.0208	0.0274
Std. Dev	1.1825	2.6142	3.5879
Max Up	11.5000	24.5000	22.0000
Max Down	-8.5000	-24.0000	-34.0000

3.3 Experimental Parameters

In setting the parameters for the NEAT system used to generate the MLPs, we considered parameter settings reported in prior applications of NEAT and supplemented this with some trial and error experimentation. We have not attempted to optimise the parameter settings but experimentation indicated that the results obtained were not hypersensitive to small changes in these settings. Table 4 lists the most important parameter settings we employed. The elitism proportion parameter was set to 20% which means that the top 20% of the population are passed on to the next generation without being altered by genetic operators. This ensures that the population does not lose high performing individuals to destructive mutation and crossover. The “min/max species threshold” settings allow the user to keep the number of species within a certain range. The upper limit was set to 10 and the lower limit was set to 6. As the population evolves, the number of neurons and connections in the average network increases according to the mutation probabilities in Table 4. Over time the level of complexity can increase to a level which brings with it major computational overhead. To combat this problem the average complexity is tracked, and once it breaks the complexity threshold the search switches from complexification to pruning mode. Pruning reduces the population’s average complexity until it falls below the specified threshold (100 in our case). This is achieved by randomly removing nodes and edges from more complex individuals.

Table 4. Experimental Parameters

Parameter	Value
Pop size	500
Mutate weights	.005
Add neuron	.01
Add connection	.1
Crossover	.9
Elitism Proportion	.2
Complexity threshold	150

3.4 Trading Methodology

In this paper we concentrate on inputs which are developed from the price time series of the futures. A total of 75 inputs are available for use by the MLPs being

evolved. The inputs were calculated by firstly taking the first differences (between two succeeding five minute blocks) of the open, high, low, close, and volume time series for each of the three futures, and then dividing each observation in each time series by their respective minimum price change, which is .01 in the Bund, and .005 in the Schatz and Bobl. This results in a series of tick changes for each future. Five lags of each series is then input to the network.

The networks are all initialised with 5% of the inputs chosen at random and a single bias node, connected to a single output node. Thus, the population of MLPs are initialised with minimal complexity. This leaves evolution to decide which of the other inputs should be switched on and the number of hidden nodes that should be added. The evolved MLPs output an integer between 0 and 1, which is post-processed using:

$$y = \begin{cases} 1 & \text{if } a \geq 0.6 \\ 0 & \text{if } 0.3 < a < 0.6 \\ -1 & \text{if } a \leq 0.3 \end{cases}$$

where y is the final network output and a is the MLP output before postprocessing. The resulting value of y is then input to the trading simulator. The trading simulator is used to evaluate the performance of a network over a given dataset. The simulator accepts three signals; buy (1), sell (-1), and do nothing (0). At any point in time the system is either long or short 1 future contract, or flat. If the trade position is long and we get a sell signal, the long position is closed out at the current market price. Conversely, if we are short and get a buy signal we close out the short position and go flat. This behavior results in a simple trading strategy where the network is allowed to stay out of the market. The maximum position is limited to a single future to make post trade analysis of results easier.

The trading simulator records the state of the system at the end of each five minute interval. A number of state variables are recorded including the realised profit and loss, the position (long/short), and any trade executed. Upon reaching the end of the dataset the simulator prints the state vector to a csv file for further analysis.

Fitness functions. The moving window approach yields an out-of-sample realised equity curve for agent A

$$R(A) = (r(A, t) : 0 \leq t < N) \quad (1)$$

where t are evenly spaced 5 minute intervals, N is the size of the equity curve, and $r(A, t)$ is the realised profit and loss (PnL) including cost deductions at interval t .

Total return. The first fitness function tested was the finishing PnL of the system $r(A, N - 1)$. This is a naive measure of fit as it ignores the equity curve up to and including $N-2$.

$$FF_1 = r(A, N - 1) \quad (2)$$

Total return minus max drawdown. A better approach is to subtract the maximum drawdown in $R(A)$ from the finishing PnL $r(A, N - 1)$. Subtracting the max drawdown will put a selection pressure on the search which favours networks that avoid devastating losses.

$$FF_2 = r(A, N - 1) - MDD(A, R(A)) \quad (3)$$

Although this measure is an improvement over total return, it ignores information from the equity curve.

Information ratio. The third fitness function we tested is a variation the mean change in return in a 5 minute period divided by the volatility of these changes. First, we calculate the first differences of the equity curve

$$RD(A) = (rd(A, t) : 1 \leq t < N) \quad (4)$$

where $rd(A, t) = r(A, t) - r(A, t - 1)$. We then divide the mean of the first differences by the standard deviation of the same series.

$$FF_3 = \frac{\bar{RD}(A)}{\sigma_{RD(A)}} \quad (5)$$

This ratio translates to the average change in the realised PnL in a 5 minute interval divided by the average deviation from this level. This fitness function favours networks with a good risk to reward relationship, thus deselecting agents with volatile performance. Unlike FF_1 and FF_2 , this metric considers the entire equity curve in its calculation. The denominator in equation 5 is the standard deviation statistic, which gives an equal weighting to positive and negative deviations from the mean.

Sortino ratio. Another risk adjusted ratio is the Sortino Ratio [12] which does not include positive deviations in its measurement of risk. This makes sense as positive volatility results in profit and should not be penalised. For the purpose of this study we define a slight variation of the original ratio

$$FF_4 = \frac{\bar{RD}(A)}{DSR(A, RD(A))} \quad (6)$$

where

$$DSR(A, RD(A)) = \sqrt{\frac{1}{N} \sum_{i=1}^n [rd(A, i) < 0][rd(A, i)]^2} \quad (7)$$

4 Results

A total of 30 runs were carried out for each of the four fitness functions discussed in Section 3.4 using the experimental parameters listed in Section 3.3. Each

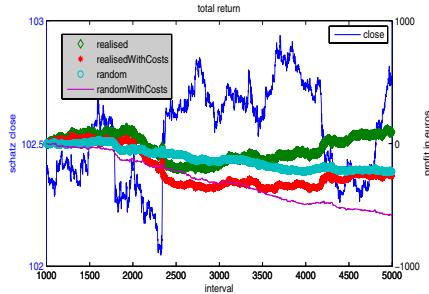


Fig. 1. Out of sample performance for fitness function 1: Total Return

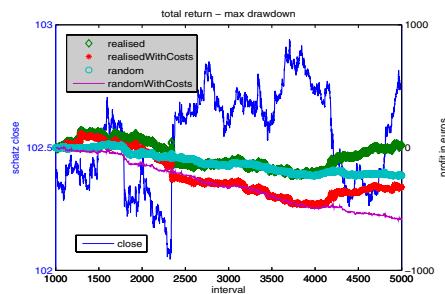


Fig. 2. Out of sample performance for fitness function 2: Total Return - Max Drawdown

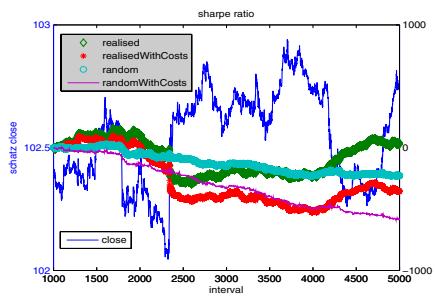


Fig. 3. Out of sample performance for fitness function 3: Info Ratio

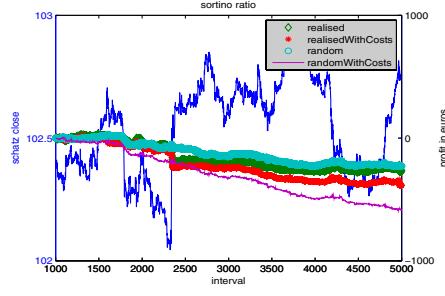


Fig. 4. Out of sample performance for fitness function 4: Sortino Ratio

run results in an out of sample equity curve for the best network, which may change between moving window increments. Figs. 1 to 4 show the equity curves for the aforementioned fitness functions averaged over the 30 runs, which are plotted against a random agent. The random agent buys, sells, or does nothing at each interval with equal probability. An average of 30 simulations is used as a benchmark.

The networks outperform the random agent both with and without transaction costs. However, the networks fail to finish in positive territory once transaction costs are included. Realistic transaction costs were used as specified by Eurex: a single contract round trip costs 40 cents (20 cents each way). As is evident from Figs. 1 to 4 the four fitness functions did not yield significantly different results. However, the trading strategy was limited to a single contract position, and it would be interesting to let the system buy/sell a variable number of contracts depending on the strength of the signal and see if the different fitness functions result in different behavior. It is possible that agents evolved with risk adjusted metrics of return as their fitness function would yield more conservative trading behavior relative to those agents evolved using fitness functions based on non risk adjusted measure of return and this will be investigated in future work.

5 Conclusion and Future Work

This study presents a novel application of a neuro-evolutionary methodology for the purposes of the intraday trading of German government bond futures. To date, few studies have examined the potential utility of computational intelligence methodologies for the purpose of trading on this market and none have adopted a neuro-evolutionary approach. In spite of restricting attention to a very limited set of potential inputs, the results suggest that the resulting model is capable of uncovering structure in the time series of bond futures prices and is capable of using this information to trade with some success. The out of sample results for the four fitness functions were not significantly different. However, due to the fact that the trading strategy was limited to a 1 lot position there was not much room for evolution to induce different levels of trading aggressiveness. An interesting piece of future work might be to allow the agents to buy/sell a variable number of contracts depending on the strength of the signal.

Of course, no conclusive assessment of the utility of this methodology can be made on the basis of the initial experiments we have undertaken in this study and we intend to pursue multiple avenues to further extend this work. For example, the current trading simulator only uses a limited range of inputs and its power could be further enhanced by use of established filter rules for price data such as technical indicators. We also note that we adopt a simple trading strategy in this study, whereby the simulator can only go long or short one contract in response to a buy/sell signal. We intend to refine this in order to allow the system to build up a larger long / short position in response to successive buy/sell signals and also to allow the system to buy/sell varying numbers of contracts depending on the strength of the signal generated by the system. We also intend to investigate the application of grammar-based GP for trading this market.

References

1. Securities industry and financial markets association - research quarterly. Technical report (November 2007)
2. Azzini, A., Tettamanzi, A.G.B.: Evolving neural networks for static single-position automated trading. *J. Artif. Evol. App.* 8(2), 1–17 (2008)
3. Bhattacharyya, S., Pictet, O.V., Zumbach, G.: Knowledge-intensive genetic discovery in foreign exchange markets. *IEEE Transactions on Evolutionary Computation* 6(2) (April 2002)
4. Brabazon, A., O'Neill, M.: Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. *Computational Management Science* 1(3), 311–327 (2004)
5. Dempsey, I., O'Neill, M., Brabazon, A.: Live trading with grammatical evolution. In: GECCO 2004 Workshop Proceedings, June 26–30 (2004)
6. Dempster, M.A.H., Jones, C.M.: A real-time adaptive trading system using genetic programming. *Quantitative Finance* 1, 397–413 (2000)
7. Dempster, M.A.H., Payne, T.W., Romahi, Y., Thompson, G.W.P.: Computational learning techniques for intraday fx trading using popular technical indicators. *IEEE Transactions on Neural Networks* 12(4), 744–754 (2001)

8. Dittmar, R., Neely, C.J., Weller, P.: Is technical analysis in the foreign exchange market profitable? a genetic programming approach. CEPR Discussion Papers 1480, C.E.P.R. Discussion Papers (September 1996)
9. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. *Journal of Financial Economics* 51, 245–271(27) (1999)
10. Neely, C.J., Weller, P.A.: Intraday technical trading in the foreign exchange market. *Journal of International Money and Finance* 22(2), 223–237 (2003)
11. Neely, C.J.: Using genetic algorithms to find technical trading rules: A comment on risk adjustment. SSRN eLibrary (1999)
12. Sortino, F.A., van der Meer, R.: Downside risk. *Journal of Portfolio Management* 17, 27–31 (1991)
13. Stanley, K., Miikkulainen, R.: Efficient evolution of neural network topologies. In: *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pp. 1757–1762. IEEE Press, Los Alamitos (2002)
14. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10, 99–127 (2002)

Decay of Invincible Clusters of Cooperators in the Evolutionary Prisoner's Dilemma Game

Ching King Chan and Kwok Yip Szeto*

Department of Physics, Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong, HKSAR, China
phszeto@ust.hk

Abstract. Two types of invincible clusters of cooperators are defined in the one-dimensional evolutionary Prisoner's Dilemma game. These invincible clusters can either be peaceful or aggressive. The survival of these invincible clusters is discussed in the context of the repeated Prisoner's Dilemma game with imitation and asynchronous updating procedure. The decay rates for these two types of clusters are analyzed numerically, for all enumeration of the configuration for small chain size. We find characteristic difference in the decay patterns of these two types of invincible clusters. The peaceful invincible clusters experience monotonic exponential decay, while the aggressive ones shows an interesting minimum in the density of cooperators before going through a slow exponential decay at long time. A heuristic argument for the existence of the minima is provided.

1 Introduction

As a key paradigm behind many scientific disciplines, game theory [1] has attracted attention of many scientists working in complex systems. One reason is that an experimental playground in computer simulation of multi-agent systems is now available [2]. One can readily test many ideas on evolution using a model for population dynamics that relates the payoff of a game to reproductive success. In numerical simulation, the agents are the players who are interacting in a given network. Since agents do not change their intrinsic nature, we can use the techniques in non-equilibrium statistical physics to describe the evolution of the population. The problem has many similar features to pattern evolution of an Ising spin system [3]. This association naturally leads to the consideration of the topology of the network of interacting agents. In real applications, the network can be a model of social contacts, while in problems of co-evolution, the topology can also be modified by the interactions [4].

Evolutionary game theory was introduced more than 30 years ago by Maynard Smith and Price [5, 6]. One of the important issues of this theory is to understand the spontaneous cooperation towards a more efficient outcome with agent interactions in the absence of a central planner [7, 8]. One popular game

* Corresponding author.

which has been studied by political scientists and sociologists is the Prisoner's Dilemma, as it provides a simple example of the difficulties of cooperation [9]. Much work on the Prisoner's Dilemma problem have been done on various networks of players playing games with neighbors in a noisy environment, and with various updating scheme for the dynamics. Nowak and co-workers have done an amazing amount of work using this model and arrive at five rules for the origin of cooperation [8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. Among the many features investigated for the Prisoner's Dilemma game (PDG), we here discuss a simple example that illustrates the importance of the topology on the survival of cooperators in a world of hostile and selfish defectors. The result may shed some lights on the problem of the interplay between dynamics and its induced modification of the topology of the network [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46].

In section 2, we introduce the parameters used in our PDG and the concept of invasion front. In section 3 we define two kinds of invincible clusters, and illustrate the possible dynamics for the expansion of these clusters. We then give the numerical results of the evolution of chains of players in the form of ring with auxiliary nodes and a ring of triangles in section 4.

2 Model of the Evolutionary Game

In PDG, two players are offered a certain reward R for mutual cooperation, and a penalty P for mutual defection. If one player cooperates while the other defects, then the cooperator gets the lowest payoff S while the defector gains the highest payoff T , with $T > R > P > S$. Game theory predicts that in Nash equilibrium, as each prisoner should minimize the maximum damage, both of them will defect. Thus, in the non-repeated Prisoner's Dilemma, defectors dominate cooperators. In the repeated PDG, however, we can find many strategies allowing cooperative behavior undefeatable by defectors as the same two players meet more than once [9, 10]. The payoff matrix for this 2-player game is given by a 2×2 matrix, denoting the two possible strategies of each player,

$$\begin{matrix} & \begin{matrix} C & D \end{matrix} \\ \begin{matrix} C \\ D \end{matrix} & \begin{pmatrix} R & S \\ T & P \end{pmatrix} \end{matrix}. \quad (1)$$

Here C stands for cooperate and D stands for defect. In this paper, we adopt the rescaled version of the interaction matrix A , by Nowak and May [13],

$$A = \begin{pmatrix} 1 & c \\ b & 0 \end{pmatrix} \text{ with } 1 < b < 2 - c, c \leq 0. \quad (2)$$

and assume that $c = 0$ which has been shown in [13] that the value of c does not affect the mechanics of the game much. Furthermore, we can rewrite this game

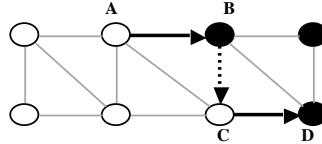


Fig. 1. Adaptive dynamics of 1D PDG. Open circles stand for cooperators and filled circles for defectors. We denote edges for cooperator adoption as a solid arrow, and defector adoption as dashed arrow.

into a two-state Potts model Hamiltonian [2, 47] to describe the total payoff of player i by

$$H_i = \sum_{j \sim i} S_i^T A S_j \text{ with } S_i, S_j \in \{\mathbf{C}, \mathbf{D}\} \text{ and } \mathbf{C} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{D} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad (3)$$

Here S_j is the state vector for the player j who is the neighbor of player i and the state vector can be one of $\{\mathbf{C}, \mathbf{D}\}$. The summation runs over all the neighbors of player i , while the neighborhood is defined by the topology of the given network. The topology we consider is a finite chain. In the actual evolution, we must also specify the updating rules. Following [14], we update by the imitation rule. Over every regular interval $\Delta t := 1$, a random node u among the whole network is chosen and set to compare with one of its competitors (neighbors), v , which is again chosen randomly. If u 's payoff is lower than v 's at that time, u will adopt v 's strategy; otherwise, nothing happens. This evolution rule is “noiseless” because u always adopt v 's strategy if it seems beneficial to u .

Players are represented by nodes on a network, and edges are built when two individuals play a game. The payoff of each node in each round is collected, and players may change their strategy using these information. The updating is asynchronous, meaning that the players modify their strategies independently of each other. With the chosen parameters, our PDG has the following simple counting: the payoff of a cooperator is simply the number of neighboring cooperators, while the payoff of a defector is b times the number of neighboring cooperators. We do not consider self-play.

In this context, we can consider the adaptive dynamics of the PDG defined in 1D. Let's illustrate the evolution with Fig. 1. If node D chooses C for comparison, D will adopt C's strategy (cooperate) because $2 > b$. The edge pointing from C to D is a solid arrow, indicating that C's strategy propagates to D and D becomes a cooperator. On the other hand, if C chooses B for comparison, C will defect because $2b > 2$. In this case, the edge pointing from B to C is a dashed arrow, meaning that B's strategy propagates to C and C will become a defector. These arrows show the propagation of strategies and their graphical interpretation defines what we call the invasion fronts. For simplicity, we assume b is just a little bit larger than 1, so $n + 1 > nb > n$ for a large enough positive integer n . Here, a 1D network with periodic boundary condition refers to a regular network that the nodes can be arranged on a ring with the links connecting nodes in the local neighborhood and have a rotational symmetry.

3 Invincible Clusters of Cooperators

From the analysis of invasion front, we expect cooperators can survive with higher chance if they group together. In 1D, we notice that there are interesting structures that we call *invincible clusters of cooperators* can resist the invasion of defectors. An invincible cluster is a subgraph consisting of only cooperators that won't defect even when placed in a sea of defectors. In the other word, an invincible cluster has no invasion fronts pointing inwards. The simplest example of an invincible cluster of cooperators is the whole network. Without any defector, the cooperators are impossible to defeat and they will survive indefinitely in a noiseless world.

If we allow evolution, an invincible cluster will naturally grow since its invasion fronts are all pointing outward. An invincible cluster that can evolve into a larger invincible cluster is called an *aggressive invincible cluster*; otherwise, we call it a *peaceful invincible cluster*. (Fig. 2)

To appreciate the meaning endowed in the names for these clusters, we show in Fig. 3 and 4 how they evolve.

In Fig. 3, the peaceful cluster tries to expand. However, as soon as it converts a neighboring defector to become a cooperator, the newly converted will be influenced by its neighbor and convert back to a defector in the following time step. We conclude that for peaceful invincible clusters, they can survive indefinitely as long as the surrounding environment has reached equilibrium, although they cannot expand.

On the other hand, in Fig. 4, the aggressive cluster can expand. Aggressiveness of cooperators is the key of counter-attacking defectors in PDG. However, when

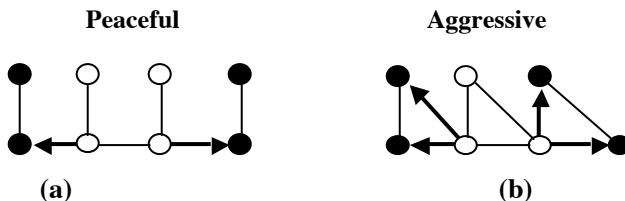


Fig. 2. Two types of invincible clusters (a) the peaceful invincible clusters in a simple ring with auxiliary nodes, and (b) the aggressive clusters in a ring of triangles

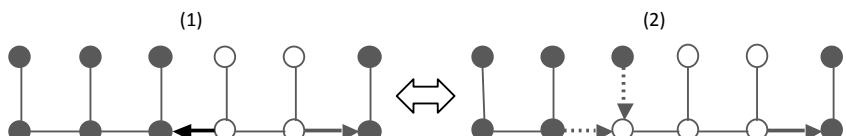


Fig. 3. A peaceful invincible cluster in (1) will expand to (2), which cannot expand further more, so it is forced to retreat to (1) again

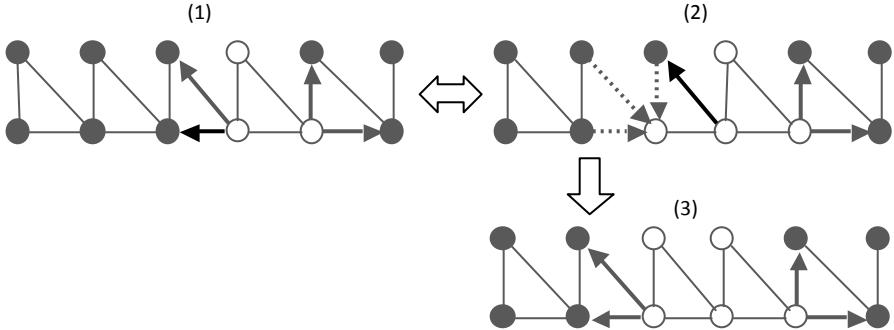


Fig. 4. An aggressive invincible cluster at step (1) will expand to (2). With 75% chance it will retreat, but there remains a 25% chance that it will further expand to (3), thereby becoming an even larger invincible cluster.

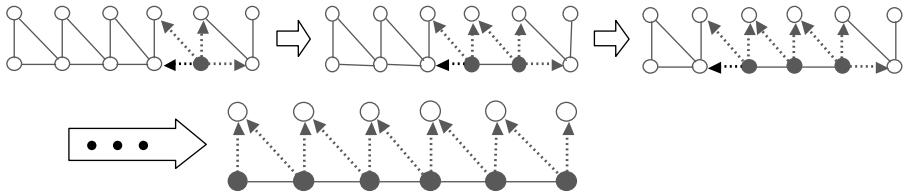


Fig. 5. Mechanism of desperate attack by a single defector that turn the entire chain of cooperators to defectors

the defectors become the minority, they will get more payoff by defecting, which may then trigger a massacre of the cooperators. This interesting feature of “*Great Revenge on the verge of Extinction*” by the defectors is illustrated in Fig. 5.

Here, we see how a single defector can turn the table over in a ring of triangles. When only one defector is left, the payoff ($4b$) is higher than its neighbors (≤ 3). If the defector decides to counter attack along the horizontal line, a larger “invincible cluster of defectors” will form. This process can continue indefinitely until all remaining cooperators are trapped in the tip of the triangles, where they can be easily converted. This is the process by which the single desperate defector can trigger the massacre of the entire army of cooperators. Although the probability of occurrence of this particular strategy for the defector is small, yet it is nonzero. Given infinite time, the event that all cooperators are trapped as shown in Fig. 5 will happen, according to Kolmogorov’s 0-1 law [48].

These discussions only address some of the important paths of evolution of configurations containing these invincible clusters. To address the equilibrium properties, we will verify our analysis by measuring the density of the cooperators at long time from numerical experiments. For a chain containing peaceful clusters, we expect that the density of cooperators should reach a nonzero constant as the network size increases. For a chain containing aggressive clusters, since the urge of expansion will lead to total destruction as we have shown in

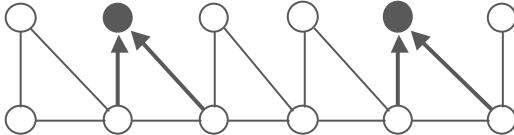


Fig. 6. An initial configuration that will lead to All-Cooperate instead of All-Defect

Fig. 5, we expect that a zeroth order estimate for the density of cooperators at equilibrium will be 2^{-N} . Of course the real situation is more complicated than this, because some initial configurations can lead to All-Cooperate instead of All-Defect, as shown in Fig. 6.

How about intermediate time scale? For peaceful clusters, the cooperator density should decay exponentially in time to a nonzero constant. If we ignore the invincible clusters for the moment, then every randomly chosen node will defect, which is exactly the theoretical analog of nuclear decay. The peaceful invincible clusters will remain and contribute to the constant value to which the exponential decay takes. For a chain containing aggressive clusters, however, such analog cannot be applied, and we anticipate more complicated behavior. Indeed, numerical simulations show that there is a dip in the density of cooperators at some finite time when there are aggressive invincible clusters in the initial configuration.

4 Evolution Computation

To quantify our claim, we computed the evolution of the two networks shown in Fig. 2 using the probability flow of all configurations in the standard analysis of Markov chain. Initially, every configuration is assumed equally probable. At each regular time interval $\Delta t := 1$, the imitation rule is carried out on every configuration to determine how much probability will be lost or gained. Because the configuration space grows as 2^N , we limit the number of nodes N to small numbers (8 to 16). Within the confine of these small set of numerical experiments, we can still observe a definitive trend from these data. The system starts its evolution with a uniform superposition of all possible configurations. It is then allowed to evolve 200 or more steps to extract the decay rate and equilibrium cooperator density ρ .

We show ρ for a chain of peaceful clusters in Fig. 7. We see that ρ at equilibrium is a rather large number, confirming our prediction that the peaceful invincible clusters can survive. ρ approaches a universal value of ~ 0.168 as $N \rightarrow +\infty$. This confirms our prediction of the equilibrium property for chain configuration of Fig. 2(a). The decay to this value is exponential at long time.

On the other hand, a ring of triangle can contain many aggressive invincible clusters (Fig. 2(b)). In this case, we expect ρ decays slowly because the aggressive invincible clusters of cooperators will deter the invasion of defectors. This is verified from our simulation. Moreover, we see from the numerical results a

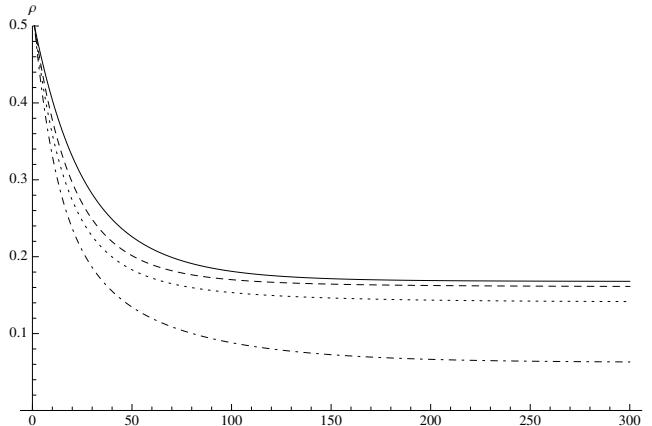


Fig. 7. Time evolution of cooperator densities of a simple ring with auxiliary nodes, with $N = 8$ (dash-dot), 10 (dotted), 12 (dashed) and 16 (solid)

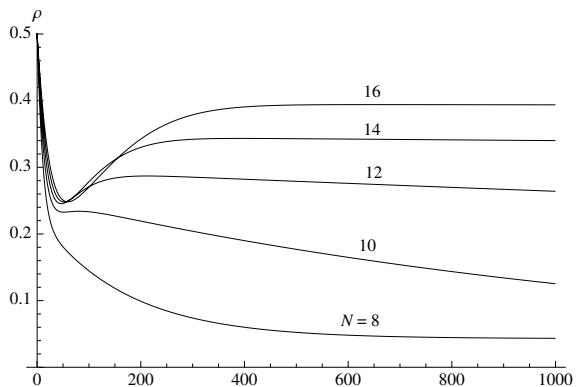


Fig. 8. Time evolution of cooperator densities of a simple ring with auxiliary nodes, with $N = 8$ to 16

rebound of cooperator density to a higher value when $\rho(t) \approx 0.25$, and then after begins the slow exponential decay. We see this interesting dip in Fig. 8 for $N > 10$. The decay for $N > 10$ is very slow, and this can effectively be treated as constant for finite time analysis.

The rebound occurs because initially the “weaker” configurations (those that cannot sustain cooperators) take time to become All-Defect, and at some turning point, the reduction of cooperators from them is taken over by the expansion of the aggressive invincible clusters. This competition between the decay of weaker configurations to All-Defect and the expansion of aggressive clusters will first decrease ρ (when the expansion loses the competition), and then rebound (when the expansion wins). However, because of the possibility of massacre by surviving

defectors, $\rho \rightarrow 0$ eventually as $t \rightarrow +\infty$. In the end, the only contribution to ρ is the configuration becoming of All-Cooperate. From the tail of the density evolution in Fig. 8 we can perform a linear regression of $\ln(\rho - \rho_\infty)$ against t to get the exponential decay rate λ . We can then extract the dependence of the decay rate with network size N , which turns out to be a power law, $\lambda \propto N^{-10.3}$.

5 Conclusions

We have demonstrated how network topology can affect evolution of cooperators in an evolutionary PDG by introducing invincible clusters of cooperators. The peaceful clusters are survivors but they cannot expand. Niche of these survivors will contribute a fixed density of cooperators in the 1D chain, and they form the sustainable resource of cooperation. The cooperator density will reach constant in equilibrium even when the network is large. Evolution pathways that lead to these peaceful clusters are therefore very important for the stability of a cooperative world. We also demonstrate another kind of invincible cluster by a ring of triangles. These aggressive clusters can expand, but due to their success, a single defector can trigger a massacre by a well defined strategy so that all cooperators are eliminated, leading to zero cooperator density in long term, although the ultra slow decay rate compensate this. Our simple analysis on the PDG in 1D, which agrees with numerical results and provides heuristic explanation, opens up many more interesting questions in higher dimension. The complex population dynamics of PDG are fascinating, and the concept of invincible clusters appears to be very useful in the understanding of the evolution of the game. For future work, we will expand our analysis to higher dimension, as well as in noisy environment, where the switching of strategy is probabilistic.

Acknowledgements. K. Y. Szeto acknowledge the support of CERG grant 602506 and 602507.

References

1. von Neumann, J., Morgenstern, O.: Theory of games and economic behaviour. Princeton University Press, Princeton (1944)
2. Szabó, G., Fath, G.: Evolutionary games on graphs. *Phys. Rep.* 446, 97 (2007)
3. Puzzo, M.L.R., Albano, E.V.: The damage spreading method in monte carlo simulations: A brief overview and applications to confined magnetic materials. *Comm. Comp. Phys.* 4, 207–230 (2008)
4. Vazquez, F., Eguiluz, V.M., Miguel, M.S.: Generic absorbing transition in coevolution dynamics. *Phys. Rev. Lett.* 100, 108702 (2008)
5. Smith, J.M., Price, G.R.: The logic of animal conflict. *Nature* 246, 15–18 (1973)
6. Smith, J.M.: Evolution and the theory of games. Cambridge University Press, Cambridge (1982)
7. Ohtsuki, H., Hauert, C., Lieberman, E., Nowak, M.A.: A simple rule for the evolution of cooperation on graphs and social networks. *Nature* 441, 502–505 (2006)

8. Nowak, M.A.: Five rules for the evolution of cooperation. *Science* 314(5805), 1560–1563 (2006)
9. Axelrod, R.: *The evolution of cooperation*. Basic Books, New York (1984)
10. Nowak, M.A.: *Evolutionary dynamics*. Harvard University Press, Cambridge (2006)
11. Nowak, M.A., Bonhoeffer, S., May, R.M.: Spatial games and the maintenance of cooperation. *P.N.A.S. USA* 91, 4877–4881 (1994)
12. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. *Nature* 359, 826–829 (1992)
13. Nowak, M.A., May, R.M.: The spatial dilemmas of evolution. *Int. J. Bifurcat. Chaos* 3, 35–78 (1993)
14. Ohtsuki, H., Nowak, M.A.: The replicator equation on graphs. *J. Theo. Bio.* 243, 86–97 (2006)
15. Nowak, M.A., Sasaki, A., Taylor, C., Fudenberg, D.: Emergence of cooperation and evolutionary stability in finite populations. *Nature* 428, 646–650 (2004)
16. Nowak, M.A., Sigmund, K.: Evolution of indirect reciprocity. *Nature* 437, 1291–1298 (2005)
17. Traulsen, A., Shores, N., Nowak, M.A.: Analytical results for individual and group selection of any intensity. *B. Math. Biol.* 70, 1410–1424 (2008)
18. Nowak, M.A., Sigmund, K.: Evolutionary dynamics of biological games. *Science* 303, 793–799 (2004)
19. Langer, P., Nowak, M.A., Hauert, C.: Spatial invasion of cooperation. *J. Theo. Bio.* 250, 634–641 (2008)
20. Nowak, M.A., Ohtsuki, H.: Prevolutionary dynamics and the origin of evolution. *P.N.A.S. USA* (to appear, 2008)
21. Ohtsuki, H., Nowak, M.A.: Evolutionary stability on graphs. *J. Theo. Bio.* 251, 698–707 (2008)
22. Pacheco, J., Traulsen, A., Ohtsuki, H., Nowak, M.A.: Repeated games and direct reciprocity under active linking. *J. Theo. Bio.* 250, 723–731 (2008)
23. Pusch, A., Weber, S., Porto, M.: Impact of topology on the dynamical organization of cooperation in the prisoner's dilemma game. *Phys. Rev. E* 77, 036120 (2008)
24. Suzuki, R., Kato, M., Arita, T.: Cyclic coevolution of cooperative behaviors and network structures. *Phys. Rev. E* 77, 021911 (2008)
25. Perc, M., Szolnoki, A.: Social diversity and promotion of cooperation in the spatial prisoner's dilemma game. *Phys. Rev. E* 77, 011904 (2008)
26. Tanimoto, J.: Dilemma solving by the coevolution of networks and strategy in a 2×2 game. *Phys. Rev. E* 76, 021126 (2007)
27. Wu, Z.X., Wang, Y.-H.: Cooperation enhanced by the difference between interaction and learning neighborhoods for evolutionary spatial prisoner's dilemma games. *Phys. Rev. E* 75, 041114 (2007)
28. Gómez-Gardeñes, J., Campillo, M., Floría, L.M., Moreno, Y.: Dynamical organization of cooperation in complex topologies. *Phys. Rev. Lett.* 98, 108103 (2007)
29. Perc, M.: Transition from gaussian to levy distributions of stochastic payoff variations in the spatial prisoner's dilemma game. *Phys. Rev. Lett.* 98, 108103 (2007)
30. Wu, Z.X., Xu, X.J., Huang, Z.G., Wang, S.J., Wang, Y.H.: Evolutionary prisoner's dilemma game with dynamic preferential selection. *Phys. Rev. E* 74, 021107 (2006)
31. Vukov, J., Szabó, G., Szolnoki, A.: Cooperation in the noisy case: Prisoner's dilemma game on two types of regular random graphs. *Phys. Rev. E* 73, 067103 (2006)
32. Zimmermann, M.G., Eguiluz, V.M.: Cooperation, social networks, and the emergence of leadership in a prisoner's dilemma with adaptive local interactions. *Phys. Rev. E* 72, 056118 (2005)

33. Vukov, J., Szabó, G.: Evolutionary prisoner's dilemma game on hierarchical lattices. *Phys. Rev. E* 71, 036133 (2005)
34. Ariosa, D., Fort, H.: Extended estimator approach for 2×2 games and its mapping to the ising hamiltonian. *Phys. Rev. E* 71, 036133 (2005)
35. Frean, M.R., Abraham, E.R.: Adaptation and enslavement in endosymbiont-host associations. *Phys. Rev. E* 69, 051913 (2004)
36. Fort, H., Viola, S.: Self-organization in a simple model of adaptive agents playing 2×2 games with arbitrary payoff matrices. *Phys. Rev. E* 69, 036110 (2004)
37. Szabó, G., Vukov, J.: Cooperation for volunteering and partially random partnerships. *Phys. Rev. E* 69, 036107 (2004)
38. Holme, P., Trusina, A., Kim, B.J., Minnhagen, P.: Prisoners' dilemma in real-world acquaintance networks: Spikes and quasiequilibria induced by the interplay between structure and dynamics. *Phys. Rev. E* 68, 030901 (2003)
39. Fort, H.: Cooperation and self-regulation in a model of agents playing different games. *Phys. Rev. E* 66, 062903 (2002)
40. Szabó, G., Hauert, C.: Evolutionary prisoner's dilemma games with voluntary participation. *Phys. Rev. E* 66, 062903 (2002)
41. Ebel, H., Bornholdt, S.: Coevolutionary games on networks. *Phys. Rev. E* 66, 056118 (2002)
42. Lim, Y.F., Chen, K., Jayaprakash, C.: Scale-invariant behavior in a spatial game of prisoners' dilemma. *Phys. Rev. E* 65, 026134 (2002)
43. Tomochi, M., Kono, M.: Spatial prisoner's dilemma games with dynamic payoff matrices. *Phys. Rev. E* 65, 026112 (2002)
44. Vainstein, M.H., Arenzon, J.J.: Disordered environments in spatial games. *Phys. Rev. E* 64, 051905 (2001)
45. Abramson, G., Kuperman, M.: Social games in a social network. *Phys. Rev. E* 63, 030901 (2001)
46. Szabó, G., Antal, T., Szabó, P., Droz, M.: Spatial evolutionary prisoner's dilemma game with three strategies and external constraints. *Phys. Rev. E* 62, 1095 (2000)
47. Szabó, G., Vukov, J., Szolnoki, A.: Phase diagrams for an evolutionary prisoner's dilemma game on two-dimensional lattices. *Phys. Rev. E* 72, 047107 (2005)
48. Brzezniak, Z., Zastawniak, T.: Basic stochastic processes. Springer, Heidelberg (2000)

Evolutionary Equilibria Detection in Non-cooperative Games

D. Dumitrescu, Rodica Ioana Lung, and Tudor Dan Mihoc

University Babeş Bolyai, Cluj Napoca, Romania

Abstract. An evolutionary approach for detecting equilibria in non-cooperative game is proposed. Appropriate generative relations (between strategies) are introduced in order to characterize game equilibria. The concept of game is generalized by allowing players to have different types of rationality. Experimental results indicate the potential of the proposed concepts and technique.

1 Introduction

Several types of equilibrium (solution) concepts have been proposed in non-cooperative Game Theory [7] [1] [3]. Each concept of equilibrium can be considered as expressing a different type of rationality. Within the present day approaches each equilibrium concept is addressed separately, meaning that in a particular game players interact accordingly to a unique equilibrium concept. Therefore only players guided by the same kind of equilibrium concept are allowed to interact. Our aim is to go beyond this paradigm in order to obtain a more realistic description.

A finite strategic game is defined by $\Gamma = ((N, S_i, u_i), i = 1, n)$ where:

- N represents the set of players, $N = \{1, \dots, n\}$, n is the number of players;
- for each player $i \in N$, S_i represents the set of actions available to him, $S_i = \{s_{i_1}, s_{i_2}, \dots, s_{i_m}\}$; $S = S_1 \times S_2 \times \dots \times S_N$ is the set of all possible situations of the game;
- for each player $i \in N$, $u_i : S \rightarrow R$ represents the payoff function.

Let $U = \{u_1, \dots, u_n\}$.

We denote by (s_{ij}, s_{-i}^*) the strategy profile obtained from s^* by replacing the strategy of player i with s_{ij} i.e.

$$(s_{ij}, s_{-i}^*) = (s_1^*, s_2^*, \dots, s_{i-1}^*, s_{ij}, s_{i+1}^*, \dots, s_n^*).$$

A strategy is called Nash equilibrium [5] if each player has no incentive to unilaterally deviate i.e. it can not improve the payoff by modifying its strategy while the others do not modify theirs. Informally we can state that these players behaviour is Nash-driven. We may also say that they play according to a Nash meta-strategy or they have a Nash type of rationality. In a similar manner we can speak about Pareto, stable state or other types of rationality.

A natural question is what happens if the players compete against each other relying on different types of rationality. A generalized game were agents are not uniform with respect to the rationality type is introduced. The type of rationality may be considered as reflecting the player interests, bias or subjectivity. For instance the players can be more or less cooperative, more or less competitive. In this way we can also allow players to be biased towards selfish or altruistic behaviour.

We assume the rationality type is described by an adequate meta-strategy concept. In a game players may assume different meta-strategies. The new paradigm offers a more realistic view and opens the possibility to further development in the Game Theory and significant applications. For instance multi-agent systems could benefit from the new approach.

The concept of generalized game with players characterized by several types of rationality is investigated. The new concepts are exemplified by considering a game were some players are Nash - and the other are Pareto-driven. An evolutionary technique for detecting the corresponding equilibrium for the generalized game is proposed.

2 Meta-strategy Concept

Consider a game with n players. Each player i has a strategy s_i consisting of m_i actions (pure strategies). Consider that each player has a certain type of rationality. Let us denote by r_i the rationality type of the player $i = 1, \dots, n$.

A meta-strategy is a system

$$(s_1|r_1, s_2|r_2, \dots, s_n|r_n),$$

where (s_1, \dots, s_n) is a strategy profile. A finite strategic generalized game is defined as a system by $G = ((N, M_i, u_i), i = 1, n)$ where:

- N represents the set of players, $N = 1, \dots, n$, n is the number of players;
- for each player $i \in N$, M_i represents the set of available meta-strategies,
- $M = M_1 \times M_2 \times \dots \times M_N$ is the set of all possible situations of the generalized game and $S = S_1 \times S_2 \times \dots \times S_N$ is the set of all strategies;
- for each player $i \in N$, $u_i : S \rightarrow \mathbf{R}$ represents the payoff function.

Remark 1. In a generalized game the set of all possible meta-strategies may also be called the meta-strategy search space.

If $r_1 = r_2 = \dots = r_n$, we say that the meta-strategy search space is *regular*. Otherwise it is said to be *irregular*.

In an irregular meta strategy space we may have different types of strategies expressing different types of rationality. It is natural to assume that the rationality type induces a particular type of equilibrium. We intend to explore the influence of irregularity on the detected equilibria. In this respect an evolutionary approach can be useful.

3 Generative Relations in Generalized Games

In order to capture several kind of equilibria we rely on particular relations between meta-strategies. The intuition behind this assumption is that each type of equilibrium can be expressed by an appropriate *generative relation*. For example if we are interested in Pareto equilibrium, standard Pareto dominance relation is considered. Mixed equilibria can be also be detected. The problem is to find the appropriate generative relation for a certain type of equilibrium.

3.1 Pareto Domination Relation

In this section three generative relations are considered. Two of them correspond to Pareto and Nash equilibria. The third induces a new type of equilibrium which we called mixed Nash-Pareto equilibrium.

Let us consider two strategy profiles x and y from S .

Definition 1. *The strategy profile x Pareto dominates the strategy profile y (and we write $x < P y$) if the payoff of each player using strategy x is greater or equal to the payoff associated to the strategy y and at least one payoff is strictly greater.*

More formal we can write $x < P y$ iff

$$u_i(x) \geq u_i(y), \text{ for each } i = 1, \dots, n,$$

and there is some index j that

$$u_j(x) > u_j(y).$$

The set of all non-dominated strategies (Pareto frontier) represents the set of Pareto equilibria of the game.

3.2 Nash – Ascendancy Relation

Similar to Pareto equilibrium a particular relation between strategy profiles can be used in order to describe Nash rationality. This relation is called Nash-ascendancy (NA).

Definition 2. *The strategy profile x Nash-ascends the strategy profile y , and we write $x < NA y$ if there are less players i that can increase their payoffs by switching their strategy from x_i to y_i then vice versa.*

Let s^* be a strategy profile. We denote by (s_{ij}, s^*_{-i}) the strategy profile obtained from s^* by replacing the strategy of the player i by s_{ij} i.e.

$$(s_{ij}, s^*_{-i}) = (s_1^*, s_2^*, \dots, s_{i-1}^*, s_{ij}, s_{i+1}^*, \dots, s_n^*)$$

In [6] is introduced an operator

$$k : S \times S \rightarrow \mathbf{N},$$

$$k(y, x) = \text{card}\{i \in \{1, \dots, n\} | u_i(x_i, y_{-i}) \geq u_i(y), x_i \neq y_i\}.$$

$k(y, x)$ denotes the number of players which benefit by switching from y to x .

Proposition 1. *The strategy x Nash-ascends y (x is NA-preferred to y), and we write $x < NA y$, if the inequality*

$$k(x, y) < k(y, x),$$

holds.

Definition 3. *The strategy profile s^* is said to be non-Nash-overcome (non-NO) in S if there is no strategy $s \in S$, $s \neq s^*$, such that $s < NA s^*$.*

According to [6] the set of all non-NO strategies from S equals the set of Nash equilibria.

This result proves that the Nash ascendancy is the generative relation for the Nash equilibrium.

3.3 NP – Efficiency Relation

Let us consider two meta-strategies

$$x = (x_1|r_1, x_2|r_2, \dots, x_n|r_n),$$

and

$$y = (y_1|r_1, y_2|r_2, \dots, y_n|r_n).$$

Let us denote by I_N the set of Nash biased players (N-players) and by I_P the set of Pareto biased players (P-players). Therefore we have

$$I_N = \{i \in \{1, \dots, n\} | r_i = \text{Nash}\},$$

and

$$I_P = \{j \in \{1, \dots, n\} | r_j = \text{Pareto}\}.$$

Let us introduce an operator E , measuring the relative efficiency of meta-strategies:

$$E : M \times M \rightarrow \mathbf{N},$$

defined as

$$\begin{aligned} E(x, y) = \text{card}(\{i \in I_N | u_i(x_i, y_{-i}) \geq u_i(y), x_i \neq y_i\} \cup \\ \{j \in I_P | u_j(x) < u_j(y), x \neq y\}). \end{aligned}$$

Remark 2. $E(x, y)$ measures the *relative efficiency* of the meta-strategy x with respect to the meta-strategy y .

The relative efficiency enables us to define a relation between meta-strategies.

Definition 4. *Let $M_1, M_2 \in M$. The meta-strategy M_1 is more efficient than meta-strategy M_2 , and we write $M_1 < E M_2$, iff*

$$E(M_1, M_2) < E(M_2, M_1).$$

Efficiency relation can be easily generalized for n , $n > 2$, types of rationality.

In what follows we consider that efficiency relation induces a new type of equilibrium called *mixed Nash-Pareto equilibrium*.

Remark 3. Nash-Pareto equilibrium defined in this section is a new concept completely different from the existing concept of Pareto-Nash equilibria [8].

4 Generalized Two Player Game

Consider a two player non-cooperative game. Let r_i be the rationality type of player i . If $r_1 = r_2 = \text{Nash}$ then both players are Nash biased and the corresponding solution concept is the Nash equilibrium.

If $r_1 = r_2 = \text{Pareto}$ then both players are Pareto-biased and the corresponding equilibria are described by the set of non-dominated strategies (Pareto front).

We intend to explore the mixed case where the first player is Nash biased and the second one is Pareto biased, i.e.

$$r_1 = \text{Nash}, r_2 = \text{Pareto}.$$

Remark 4. In this case the sets I_N and I_P form a partition of the set $\{1, \dots, n\}$.

In order to detect the mixed Nash-Pareto equilibria of the generalized game an evolutionary approach may be used.

4.1 Detecting Mixed N-P Equilibria in Generalized Games

Proposed technique for detecting equilibria in (generalized) games evolves a population of meta-strategies

Let us consider an initial population $P(0)$ of p meta-strategies for the generalized two player game. Each member of the population has the form

$$x = (s_1|r_1, s_2|r_2).$$

Pairs of meta-strategies are randomly chosen from the current population $P(t)$. For each pair a binary tournament is considered. The meta-strategies are compared by means of the efficiency relation. An arbitrary tie breaking is used if the two meta-strategies have the same efficiency. The winners of two binary tournaments are recombined using the simulated binary crossover (SBX) operator [9] resulting two offspring. Offspring population is mutated using real polynomial mutation [2], resulting an intermediate population P' . Population $P(t)$ and P' are merged.

The resulting set of meta-strategies is sorted with respect to the efficiency relation using a fast non-dominant sorting approach similar with [2]. For each meta-strategy M' the number expressing how many meta-strategies in the merged population are less efficient than M' is computed. On this basis the first p meta-strategies are selected from the merged population. Selected meta strategies represent the new population $P(t+1)$.

Let us remark that in the proposed technique selection for recombination and survival is driven by the efficiency relation. Therefore the population of meta-strategies is expected to converge towards the mixed Nash Pareto front. According to the proposed approach the members of this front represent the N-P equilibria of the generalized game.

The previous method is called *Relation based Equilibrium Detection (RED) Algorithm* and can be described as follows:

RED algorithm

- S1. Set $t = 0$;
- S2. Randomly initialize a population $P(0)$ of meta-strategies;
- S3. Binary tournament selection and recombination for $P(t) \rightarrow Q$;
- S4. Mutation on $Q \rightarrow P'$;
- S5. Compute the rank of each population in member $P(t) \cup P'$ with respect to the efficiency relation. Order by rank $(P(t) \cup P')$;
- S6. Rank based selection for survival $\rightarrow P(t + 1)$;
- S7. Repeat steps S3 - S6 until the maximum generation number is reached.

5 Numerical Experiments

In order to exemplify the proposed concepts the duopoly Cournot model is considered (see for instance [4]).

Let q_1 and q_2 denote the quantities of an homogeneous product - produced by two companies respectively. The market clearing price is

$$P(Q) = aQ,$$

where

$$Q = q_1 + q_2,$$

is the aggregate quantity on the market. Hence we have

$$P(Q) = \begin{cases} a - Q, & \text{for } Q < a, \\ 0, & \text{for } Q \geq a. \end{cases}$$

Let us assume that the total cost for the company i of producing quantity q_i is $C_i(q_i) = cq_i$. Therefore, there are no fixed costs and the marginal cost c is constant, $c < a$. Suppose that the companies choose their quantities simultaneously. The payoff for the company i is its profit, which can be expressed as:

$$\begin{aligned} \pi_i(q_i, q_j) &= q_i P(Q) - C_i(q_i) \\ &= q_i [a - (q_i + q_j) - c]. \end{aligned}$$

Several experiments have been performed for this game by using RED technique. The Cournot model with parameter $a = 24$ and $c = 9$ is considered in these numerical experiments.

Experiment 1. Nash - Nash rationality Let us consider two players having the same rationality type $r_1 = r_2 = \text{Nash}$. In 10 runs the RED algorithm detects the Cournot's game unique Nash equilibrium point (5,5) and the corresponding payoff (25,25). The final population is depicted in Figure 1.

Experiment 2. Pareto - Pareto rationality For $r_1 = r_2 = \text{Pareto}$ the RED population converges towards the Pareto front in 10 runs. The final population is depicted in Figure 1.

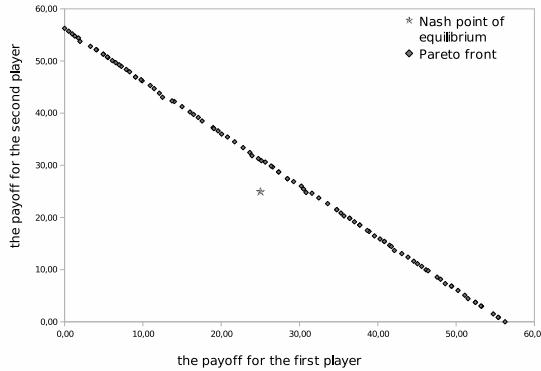


Fig. 1. Detected Nash point of equilibrium and Pareto front in 10 runs

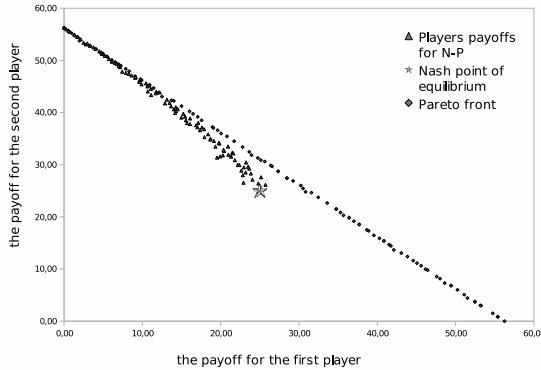


Fig. 2. Nash - Pareto front detected in 15 runs

Experiment 3. Nash - Pareto rationality. Let us now consider the players having different kind of rationality, namely $r_1 = \text{Nash}$ and $r_2 = \text{Pareto}$. In 15 runs the population $P(t)$ of meta-strategies converges towards the front depicted in Figure 2. The set of payoff values are distributed from the payoffs of the Pareto front to the payoffs corresponding to the Nash equilibrium.

Experiment 4. Pareto - Nash rationality Let us now consider the case $r_1 = \text{Pareto}$ and $r_2 = \text{Nash}$. In 15 runs the population $P(t)$ of meta-strategies converges and the set of payoff values are distributed from the payoffs corresponding to the Nash equilibrium to the payoffs of the Pareto front.

In order to have a complete picture of detected equilibria, the results are represented together in Figure 3. An expected symmetry of the detected N-P and P-N fronts with respect to the first bisector is noted. The sets of detected strategies supply a discrete representation of the mixed Nash - Pareto equilibrium.

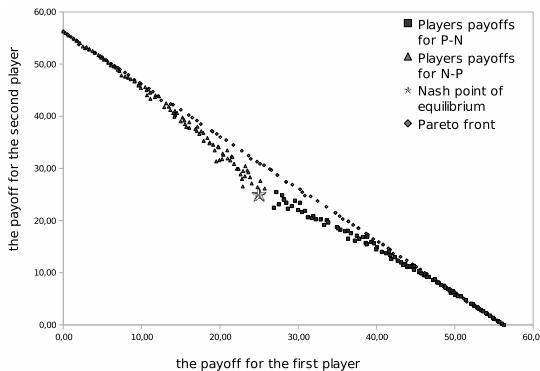


Fig. 3. Discrete representation of the pure and mixed equilibria in 15 runs

Table 1. Average payoff and standard deviation in the final population of 100 meta-strategies after 30 generations

Meta-strategy type							
Nash - Pareto		Pareto - Nash		Nash - Nash		Pareto - Pareto	
Player rationality	Avg. payoff (St.Dev.)						
Nash	13.5 (7.36)	Nash	12.5 (7.28)	Nash	25.00 (0.08)	Pareto	27.60 (16.8)
Pareto	41.00 (8.81)	Pareto	42.2 (8.71)	Nash	25.00 (0.08)	Pareto	28.50 (16.8)
Total Payoff							
	54.50		54.70		50.00		56.10

The average payoffs in the final population is given in Table 1 and Table 2. Some interesting remarks can be drawn from these tables:

- (i) the Pareto players in the final population have better average payoffs in each game (respectively the Nash players in the final population obtain poorer payoffs in each game);
- (ii) the Pareto players take an advantage in the mixed meta-strategies (N-P, P-N) compared with (P-P) meta-strategy;
- (iii) the Nash player are clearly handicapped in the mixed equilibria (meta-strategies) compared with the Nash-Nash pure equilibrium (meta-strategy);

Table 2. Average payoff and standard deviation in the final population of 200 meta-strategies after 100 generations

Meta-strategy type							
Nash - Pareto		Pareto - Nash		Nash - Nash		Pareto - Pareto	
Player	Avg. payoff (St.Dev.)	Player	Avg. payoff (St.Dev.)	Player	Avg. payoff (St.Dev.)	Player	Avg. payoff (St.Dev.)
Nash	12.86 (7.53)	Nash	12.98 (7.40)	Nash	25.00 (0.00)	Pareto	28.27 (16.4)
Pareto	41.76 (9.19)	Pareto	41.49 (8.94)	Nash	25.00 (0.00)	Pareto	27.91 (16.4)
Total Payoff							
	54.47		54.62		50.00		56.19

- (iv) The average payoff in mixed equilibria (N-P, P-N) is between those of the pure equilibria (N-N, P-P);
- (v) experiments cannot differentiate between the payoffs associated to N-P and P-N mixed equilibria.

6 Conclusions and Future Work

A concept of generalized game, involving several types of rationality is proposed. The type of rationality is captured by the concept of meta-strategy. The corresponding solution concept (equilibrium) is induced by a generative relation between meta-strategies. This allows the combination of different types of equilibria in a game.

An evolutionary technique for detecting an approximation of the generalized equilibria is developed. The idea are exemplified for a Cournot game with two types of rationality.

Experimental results offer an inside view of the problems arising when two different type of equilibria are considered in the same game. Results also indicate the potential of the proposed technique. Future work will address generalized games having more than two rationality types and more players.

References

1. Bade, S., Haeringer, G., Renou, L.: More strategies, more Nash equilibria, Working Paper 2004-15, School of Economics University of Adelaide University (2004)
2. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)

3. McKelvey, R.D., McLennan, A.: Computation of equilibria in finite games. In: Amman, H.M., Kendrick, D.A., Rust, J. (eds.) *Handbook of Computational Economics*. Elsevier, Amsterdam (1996)
4. Lung, R.I., Muresan, A.S., Filip, D.A.: Solving multi-objective optimization problems by means of natural computing with application in finance. In: *Aplimat 2006*, Bratislava, pp. 445–452 (February 2006)
5. Nash, J.F.: Non-cooperative games. *Annals of Mathematics* 54, 286–295 (1951)
6. Lung, R.I., Dumitrescu, D.: Computing Nash Equilibria by Means of Evolutionary Computation. *Int. J. of Computers, Communications & Control*, 364–368 (2008)
7. Osborne, M.J., Rubinstein, A.: *A Course in Game Theory*. MIT Press, Cambridge (1994)
8. Maskin, E.: The theory of implementation in Nash equilibrium: A survey. In: Hurwicz, L., Schmeidler, D., Sonnenschein, H. (eds.) *Social Goals and Social Organization*, pp. 173–204. Cambridge University Press, Cambridge (1985)
9. Deb, K., Beyer, H.: Self-adaptive genetic algorithms with simulated binary crossover. *Complex Systems* 9, 431–454 (1995)

Coevolution of Competing Agent Species in a Game-Like Environment

Telmo Menezes and Ernesto Costa

Centre for Informatics and Systems of the University of Coimbra
`{telmo,ernesto}@dei.uc.pt`

Abstract. Two species of agents coevolve in a 2D, physically simulated world. A simple fitness function rewards agents for shooting at agents of the other species. An evolutionary framework consisting of the grid-brain agent controller model and the SEEA steady-state evolutionary algorithm is used. We were able to observe a phenomenon of species specialization without the need for geographical separation. Species with equal initial conditions were shown to diverge to different specialization niches by way of the systems dynamics. This kind of research may lead to more interesting gaming environments, where the world keeps changing and evolving even in the absence of human interaction.

1 Introduction

The evolution of agent controllers has promising applications in the generation of computational intelligence for video games. The behaviors of computer controlled entities may adapt to the demands of the environment, instead of being explicitly programmed. This kind of approach fits particularly well with games genres that are bases on multi-agent environments. NERO [1], for example, is a recent experimental computer game that explores the evolution of agent controllers. NERO simulates a battle of agent teams, and the human player is required to train a team formed of neural network controlled agents under an evolutionary process. The human player must create training situations by placing agents and objects in the scenario, as well as tweaking a fitness function throughout the process.

We experiment with the evolution of game agents without human intervention. Our goal is to demonstrate that interesting behaviors can arise from the coevolution of competing species, using a predefined, simple fitness function.

The experimentation we performed uses gridbrains as agent controllers, and the simulation embedded genetic algorithm (SEEA) to provide the evolutionary process. Both these models have been developed to define an evolutionary framework of computational intelligence for multi-agent simulations. We encourage the reader to refer to previously published work on these models [2] [3].

In the next two section we will give an overview of the gridbrain model and the SEEA algorithm. Following, we will describe the experimental settings and then present results.

2 The Gridbrain

The gridbrain is a virtual machine designed to serve as a brain for an autonomous agent. It belongs to the family of genetic programming, with some similarities to Parallel Distributed Genetic Programming [4] [5] and Cartesian Genetic Programming [6][7]. It consists of a network of computational components placed on rectangular grids. There are two types of grid: alpha and beta. Alpha grids are associated with sensory channels and are responsible for processing perceptual information. The beta grid receives inputs from the alpha grids and outputs decisions. A gridbrain can have any number of alpha grids (one of each sensory channel), but only one beta grid.

Components are information processing units. They are the computational building blocks of gridbrains. Much like machine code instructions, components belong to classes of functionalities: input/output, arithmetic, boolean logic, information aggregation, synchronization and memory.

Components and connections may be active or inactive, depending on if they belong to an *active path*. An active path is a sequence of connections that links a component that generates information to a component that triggers and action.

The gridbrain model provides a set of genetic operators that can be used by an evolutionary algorithm to produce replication with change. There are three types of operators. Two of them are usual in evolutionary computation systems: mutation and recombination. The third, formating, deals with adapting the shape of the grids as evolution progresses and is related to the complexification of gridbrains.

Mutation operators are defined at connection level and component level. There are two pairs of connection level operators: *add/remove* and *split/join*. The operators in each pair are symmetrical, one performing the inverse operation of the other. The add operator inserts a new valid connection and the remove operator deletes an existing connection from the gridbrain. These mutations occur with respective probabilities of p_a and p_r per existing connections in the gridbrain. The split operator routes an existing connection through an intermediary component in the grid, and the joint operator reverts a previous splitting. Splits and joins occur with respective probabilities of p_s and p_j per existing connection.

There is one component level operators: *change inactive component*, which replaces an existing component with a new one. The new component is randomly selected from the grid component set. These mutations occur with a probability of p_c per component in the gridbrain.

The recombination operator for the gridbrain has to deal with the recombination of differently shaped grids, containing networks with different topologies. Furthermore, there are different types of components, so the network nodes are heterogeneous. It uses a connections tagging mechanism to use equivalent connection groups as the units of recombination. This recombination operator always produces valid gridbrains and is able to recombine functionalities in a meaningful way.

Formating is an operator that adapts the shape of the grids according to the network contained in the gridbrain. It is of a non-stochastic nature and can

be seen as an adaptation mechanism. The changes it performs do not affect the phenotypical expression of the individual. The purpose of formating is to regulate the search space of the evolutionary process. It is part of the complexification process. We attempt to create systems that are initialized with empty brains, and that undergo an increase in complexity as higher quality solutions are found. Solutions are constructed through iterated tweaking, in a similar fashion to what happens in nature. Our goal is to have the size and complexity of the gridbrains to be determined by the demands of the environment.

The set of gridbrain operators were also designed with the goal of bloat control [8], and have been shown to be effective in this respect [2] [3].

3 Simulation Embedded Evolutionary Algorithm

The *Simulation Embedded Evolutionary Algorithm* (SEEA) is a steady-state evolutionary algorithm that we developed for the purpose of evolving agents in continuous simulations without generations. It allows the seamless integration of an evolutionary process with a multi-agent simulation. New individuals may be generated at any moment and on demand.

SEEA keeps a buffer of individuals for each species in the simulation. Each time an individual dies or is removed from the population, its fitness is compared to the fitness of a random individual in the buffer. If the fitness is equal or greater, the recently removed individual replaces the old one in the buffer, otherwise the fitness of the old individual is updated using the expression:

$$f_{new} = f_{old} \cdot (1 - a)$$

where a is the *fitness ageing factor*.

The purpose of fitness ageing is to maintain diversity in the buffer. Individuals that make it to the buffer have a chance of producing offspring. The higher their fitness, the more offspring they are likely to produce, but eventually they will be replaced, even if it is by a lower fitness individual. Fitness ageing takes place when an individual in the buffer is challenged by an individual removed from the population, so it adapts to the current rate of individual removal, which can be variable and unpredictable.

When the simulation requests a new individual, two elements in the buffer are selected at random and recombined if recombination is to be done, or one is selected at random and cloned for simple mutation. Mutation is applied, followed by formating, and finally the placement of the offspring in the simulation environment. A probability of recombination, p_{rec} , is defined. This probability is used to randomly decide in each individual request if recombination is to be applied.

In simulations with fixed populations, as the one presented in this paper, agent removal is always followed by the creation of a new agent.

The basic SEEA algorithm presented is capable of creating evolutionary pressure so that, as the simulation advances, agents tend to increase their fitness value. One limitation of it is that it only promotes the emergence of behaviors

that benefit each agent directly. It may be insufficient to promote the emergence of collective behaviors, where some degree of altruism is needed.

Group fitness in SEEA is a mechanism to reflect the success of other members of a species that coexisted with an agent in the simulation on the agent's own fitness value. The principle is to allow an agent to increase its fitness by developing behaviors that benefit the fitness of other agents of its species. It is inspired on the biological theory of *group selection* [9].

The simulation environment and fitness evaluation do not have to be altered in any way for group fitness to be applied. Instead of directly using the individual fitness evaluation provided by the environment, f_i , a *composite fitness* is used. The composite fitness, f_c is calculated by the expression:

$$f_c = (1 - g) \cdot f_i + g \cdot f_g$$

where f_g is the *group fitness component* and g is the *group factor*. The group factor is a real value in the $[0, 1]$ interval. The higher it is, the more important the group success is to the composite fitness of each agent.

The group fitness component reflects the variation in the fitness of other agents, during the lifetime of the agent for which it is being calculated. The principle is to only reflect fitness variations that the agent may have helped cause in others. An effective way to compute the group fitness component is to maintain a group fitness sum, G for each agent. When an agent is sent into the environment, its G is initialized by applying the expression:

$$G = - \sum_{a \in S(t_0)} f_a(t_0),$$

where t_0 is the simulation time at which the agent was created, $S(t_0)$ is the set of agents in the world belonging to the same species as the agent we are calculating G for, at simulation time t_0 , and $f_a(t_0)$ is the current individual fitness for agent $a \in S(t_0)$. Then, during the lifetime of the agent, each time another agent of the same species dies, we increment G by that agent's final individual fitness. When an agent dies, its final group fitness sum is calculated by applying the expression:

$$G' = G + \sum_{a \in S(t)} f_a(t)$$

This way, in the end of the agent lifespan, G contains the summation of the variations of individual fitnesses in other agents of the same species, during that lifespan. Finally, the group fitness component is given by:

$$f_g = \frac{G}{pop - 1}$$

where pop is the population size of the agent's species in the environment. In case this population is variable, an average can be used. This way, f_g gives us the individual fitness variation per other agent in the environment.

4 Experimental Setup

Experimentation is done with LabLOVE [10], a tool that we developed for our research and that is available to the community as open source. LabLOVE includes an implementation of the Gridbrain, the SEEA algorithm and a physically simulated multi-agent environment, as well as real-time visualization and data recording modules.

We define a videogame inspired scenario where agents are rewarded for shooting at agents of the other species. Agents have the ability to move in a 2D physically simulated world, shoot, eat food items and emit sounds. They have two sensory channels: vision and audition. Vision provides them with information about objects in the environment, including other agents and the targets. Audition allows them to perceive sounds emitted by other agents and by shots. The effectiveness of a laser shot is related to the amount a time an agent spends aiming at a target. A simulation parameter named *laser_interval* (l_i) establishes the number of simulation cycles that an agent must spend aiming at a target for it to have full effect. The amount of damage that a target suffers from a shot is determined by the expression $d = l_i \cdot l_t \cdot d_{max}$, where d is the damage, l_t is the time the agent spent aiming at that object and d_{max} is the maximum damage that a shot can cause. Agents can increase their chances of destroying a target by cooperating, which means, shooting at the same target at the same time.

Two species are evolving at the same time, and the world is constantly populated with a fixed number of food items that the agents can consume to increase their energy level. The only difference between the two species is their color, species A being red and species B being blue. Food items always provide an increase in energy level of 1.

Gridbrains are configured to have three grids: one alpha grid to process vision sensory information, another alpha grid to process audition sensory information and the beta grid. Additionally to generic gridbrain components, input and output components of the types detailed in table 1 are included in the respective grids component sets.

The input value of an action component, i determines the intensity with which the action is triggered. The force applied in go and rotate actions, as well as the intensity of the laser shot is proportional to i . The energy costs of actions is proportional to i .

Every time an agent shoots at a target, a score is calculated by multiplying the damage caused by this shot with the number of simultaneous successful shots from other agents. We define the fitness function as the best score obtained during the agent's lifetime. This encourages the agents to evolve mechanisms to produce more effective shots, but also to synchronize their shooting with the other agents.

Evolutionary parameters are configured as following: add/remove connection probability is set to $p_a = p_r = 0.01$; split/join connections probability is set to $p_s = p_j = 0.01$; the change inactive component operator is used, with $p_c = 0.2$; change parameter is set to $p_p = 0.01$, $\delta = 1.0$; recombination probability is set to $p_{rec} = 0.25$; SEEA buffer size is set to $s_{buf} = 100$ and the fitness ageing

Table 1. Sensory inputs and actions used in the experiment

Vision Input	Description
Position	Position of the object relative to the agent's vision field. −1 is the farthest possible to the left and 1 is the farthest possible to the right.
Distance	Distance from the object, divided by its view range, resulting in a value in [0, 1].
Target	1 is object is on target, 0 otherwise.
Eating target	1 is object is current eating target, 0 otherwise.
Line of Fire	1 is currently on the target of this object, 0 otherwise.
Color	Distance between agent's color and this object's color.
Sound Input	Description
Position	Position of sound origin relative to agent.
Distance	Distance of sound source, divided by sound range.
Action	Description
Go	Apply forward force to agent.
Rotate	Apply rotation torque to agent.
Fire	Fire laser.
Eat	Attempt to eat nearest object.
Speak	Emit a sound.

factor is set to $a = 0.5$. These values were found to be good choices by previous benchmarking.

The simulation is defined to have 20 agents of each species and 20 food item. Agents have a random maximum lifespan of 9.5 to 10.5 Kcycles. They can die because this lifespan is reached, or by expending too much energy or being shot. The complete experiment definition can be found in the *battle.lua* file, that is in the *experiments* directory of the LabLOVE release.

5 Results

Four simulation runs were performed. We will present and analyze the evolutionary history of two of them.

In figure 1 we present the evolution of several metrics for the first simulation run. Each panel presents the evolution of a metric for the two species. Species A is represented in solid line, while species B is represented in dotted line. In panel a) it can be observed that both species go through a period of fitness increase until about 1×10^5 Kcycles. From this moment on, a sharp increase in the fitness of species B affects the performance of species A, causing a decline in its fitness. As can be observed in panel c), this relates to species B gaining the ability to destroy species A agents, and thus causing an increase in the species A death rate. Until the end, the simulation run goes through periods of convergence and divergence in the fitness of both species. Almost symmetrical fitness curves are created, with the axis of symmetry being an horizontal line coinciding approximately with the

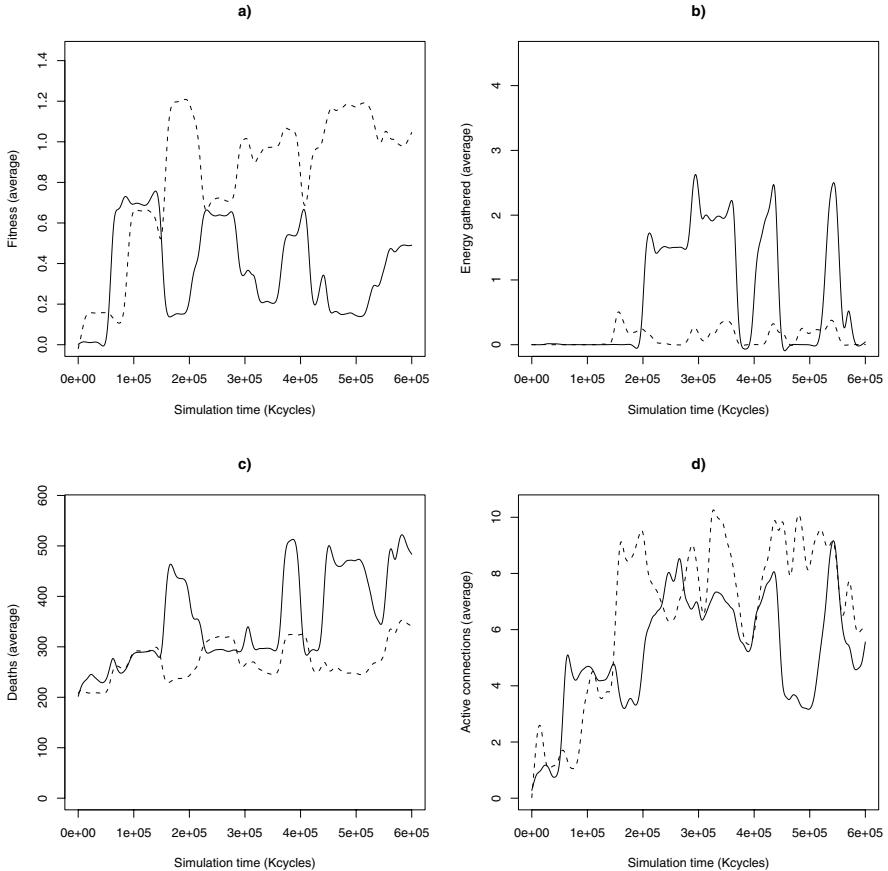


Fig. 1. Evolutionary history of a battle simulation run. Several metrics are presented for the two species. Species A is represented in solid line, species B in dotted line. a) Average fitness; b) Average energy gathered; c) Deaths; d) Average active connections.

0.6 fitness level. At several moments the fitness of both species becomes almost the same, only to diverge again, with species B always taking the lead.

In panel b) it can be observed that, for several times, species A shows the ability to gather energy by consuming food items. This is an interesting result because the consumption of food items is not directly rewarded by the fitness function. It helps indirectly, because by sustaining higher energy levels, agents are harder to destroy, and by living longer they have a greater chance of producing good shots.

In this simulation run, the two species develop different strategies. Species B relies more on its ability to destroy opponents, while species A relies more on increasing its survival chances by consuming food items. This is interesting because both species are defined with the same initial conditions. It is an indica-

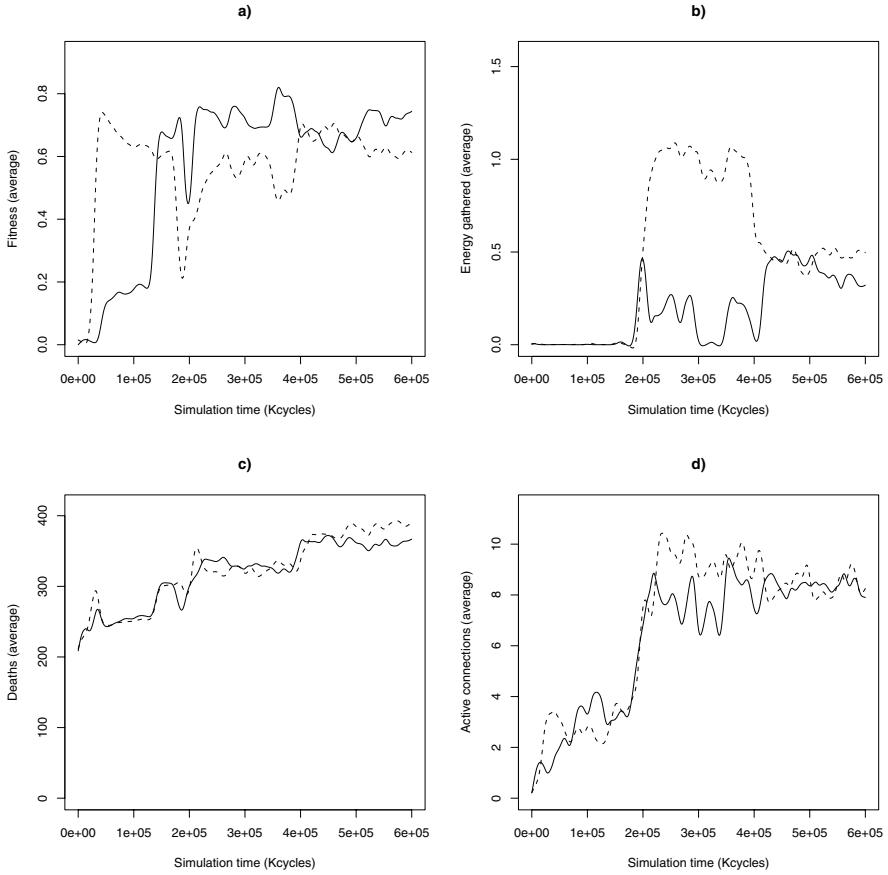


Fig. 2. Evolutionary history of another battle simulation run. Several metrics are presented for the two species. Species A is represented in solid line, species B in dotted line. a) Average fitness; b) Average energy gathered; c) Deaths; d) Average active connections.

tion of the possibility of generating diversity through species coevolution in this type of simulation. It is likely caused by initial, random changes, sending each species in a different evolutionary path. This is consistent with non-linear, complex system behavior. Also, it is a niching phenomenon caused, not by physical separation, but by the dynamics of the system.

In panel d) we can observe that average gridbrain complexity is more unstable than in previous scenarios. Species go through stronger fluctuations in the average number of active connections. We believe this is cause by the *arms race* [11] between the two species leading to a more dynamic environment.

In figure 2 we present the evolutionary history of another run for the same scenario. The reason we do it is to illustrate that this scenario leads to diverse

evolutionary histories. In this second run, it can be observed that the species attain a similar capability of destroying each other, leading to similar fitness curves oscillating around the 0.6 fitness value. The death count average for both similar also increases in a similar fashion and overall there is equilibrium between the species. As can be seen in panel b), species B goes through a period of greater ability in consuming food items, but to the end both species stabilize their consumption level around similar values.

6 Final Remarks

We were able to observe a phenomenon of species specialization without the need for geographical separation. Species with equal initial conditions were shown to diverge to different specialization niches by way of the systems dynamics. Another observation we made in this scenario is the unpredictable nature of evolutionary runs.

Coevolution under our framework shows the potential for generating diverse and surprising agent behaviors, by exploring the interaction between the fitness function, the environmental basic rules and the dynamics of coevolution of species.

This kind of research may lead to more interesting gaming environments, where the world keeps changing and evolving even in the absence of human interaction.

Acknowledgments

The first author would like to acknowledge grant SFRH/BD/19863/2004 from *Fundação para a Ciência e Tecnologia (FCT)*, Portugal.

References

- Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Evolving Neural Network Agents in the Nero Video Game. In: Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (2005)
- Menezes, T., Costa, E.: Artificial Brains as Networks of Computational Building Blocks. In: Proc. of the 5th European Conference on Complex Systems, Jerusalem, Israel (2008), <http://www.jeruccs2008.org/node/606>
- Menezes, T.: Evolutionary Computational Intelligence for Multi-Agent Simulations. Departamento de Engenharia Informatica, Faculdade de Ciencias e Tecnologia, Universidade de Coimbra (2008)
- Poli, R.: Parallel Distributed Genetic Programming. technical report CSRP-96-15. The University of Birmingham, UK (1996)
- Poli, R.: Optimization, Advanced Topics in Computer Science. In: Corne, D., et al. (eds.) Parallel Distributed Genetic Programming, ch. 27, pp. 403–431. McGraw-Hill, Maidenhead (1999)

6. Miller, J.F.: An Empirical Study of the Efficiency of Learning Boolean Functions using a Cartesian Genetic Programming Approach. In: GECCO 1999: Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, pp. 1135–1142. Morgan Kaufmann, San Francisco (1999)
7. Miller, J.F., Thomson, P.: Cartesian Genetic Programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
8. Langdon, W.B.: The Evolution of Size in Variable Length Representations. In: 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, pp. 633–638. IEEE Press, Los Alamitos (1998)
9. Smith, M.J.: Group Selection and Kin Selection. *Nature* 201, 1145–1147 (1964)
10. Menezes, T.: Lablove - Laboratory of Life on a Virtual Environment (2007), <http://sourceforge.net/projects/lablove>
11. Dawkins, R., Krebs, J.: Arms Races Between and Within Species. *Proceedings of the Royal Society of London* 205(1161), 489–511 (1979)

Simulation Minus One Makes a Game

Noriyuki Amari and Kazuto Tominaga

School of Computer Science, Tokyo University of Technology
1401-1 Katakura, Hachioji, Tokyo 192-0982 Japan
ronsum@mozart.tomilab.net, tomi@acm.org
<http://www.tomilab.net/>

Abstract. This paper presents a way to develop a game using an artificial chemistry. An artificial chemistry is an abstract model of chemical system. It is used in the research field of artificial life. We develop a roguelike game using an artificial chemistry with a specific approach, which we propose in this paper: first, we build a system to simulate the world of a roguelike game; then we remove a part of the system to make it a game. A small set of rules in the artificial chemistry is able to define the simulation, and removing a rule makes it a game. This shows the effectiveness of the present approach in developing a certain type of game using the artificial chemistry.

1 Introduction

Artificial life (ALife) is a research area where the simulation and synthesis of living systems are studied. Recently it is getting close relation to games. For example, Spore [1] is a game to create a virtual space ecosystem, and Panic Shooter [2] is a shooting game implemented on cellular automata. Since a main focus of ALife research is to give rise to life-like behaviour in computer simulation, it is suitable to produce unexpected but natural events in games [3], which have been conventionally programmed algorithmically in game software.

Artificial chemistries (AChems) are one of the research methodologies used in ALife studies [4]. An AChem is a formal model that describes an abstract chemical system. We proposed an artificial chemistry that represents molecules with character strings and that deals with compartments separated by membranes [5]. A chemical reaction in this AChem is a recombination of strings defined as a rule similar to a chemical equation. The system of rules is executed on simulation software (a simulator) of the AChem, and its behaviour is observed.

We develop a roguelike game using the AChem. A roguelike is a genre of game; the player's goal in a roguelike game is to reach the deepest part of a dungeon where monsters prowl. The player character's encounter with a monster is regarded as the collision of molecules. In this paper, we present a method to develop a game, in which we take the following two steps. First, we build a system to simulate the world of the roguelike game. Then, we remove a part of it to make it a game.

The organisation of the paper is as follows. Section 2 briefly explains the AChem. Section 3 illustrates the construction of a system that simulates the

world of a roguelike game. In Section 4, we change it into a game. Section 5 discusses the effectiveness of the approach. And Section 6 concludes the paper.

2 Artificial Chemistry with Compartments

In this section, we briefly explain an AChem with membrane connection [5], which is an extension to an AChem with membrane division and merger [6]. The base AChem [6] is explained through Sec. 2.4; Section 2.5 explains the extension.

2.1 Virtual Atoms and Virtual Molecules

A *virtual atom* (or *v-atom* for short) corresponds to an atom in nature. A v-atom is denoted by a capitalised word such as `Abc1`. A *virtual molecule* (*v-molecule*) is a string of v-atoms or a stack of such strings (called *lines*). Figure 1 illustrates an example v-molecule; it is denoted by `0#AbcDef/1#GhiJkl/`. The slashes delimit the lines, and the number at the beginning of each line is the *displacement* of the line relative to the first line. The displacement 1 of the second line means that it starts at the position to the right by one v-atom from the first line's starting position, as shown in the figure. A displacement can be negative, in which case the line starts at some position to the left from the first line.

<code>Abc</code>	<code>Def</code>
<code>Ghi</code>	<code>Jkl</code>

Fig. 1. An example v-molecule

2.2 Patterns and Wildcard Expressions

In this AChem, chemical equations are expressed in terms of *patterns*. A pattern matches (or does not match) a v-molecule.

A pattern is composed of *literals* and/or *wildcard expressions*. A literal matches a v-atom, and is denoted by the v-atom's name; for example, the literal `Abc` matches a v-atom `Abc`. There are two types of wildcard expressions. One is an *atomic wildcard expression*, denoted by a number surrounded by angle brackets such as `<1>`. An atomic wildcard expression matches any v-atom. The number is the identifier of the expression. The other is a *sequence wildcard expression*, denoted by a number and an asterisk surrounded by angle brackets such as `<*1>` and `<2*>`. A sequence wildcard expression matches any sequence of v-atoms of any length, including the null sequence. An asterisk represents the direction in which the sequence can extend.

A pattern matches a v-molecule if and only if each line of the pattern matches the corresponding line of the v-molecule with the proper displacement. For example, the pattern `0#<*1>AbCd<2*>/0#Ef<3*>/` matches the v-molecules `0#AbCd/0#Ef/`, `0#AbCd/0#EfGh/` and `0#YzAbCd/1#Ef/` (Fig. 2). Note that, in the last case, the displacement of the second line is 1 since the expression `<*1>` represents `Yz` to make the position of `Ab` (and thus of `Ef`) become 1.

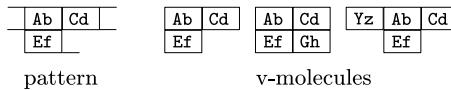
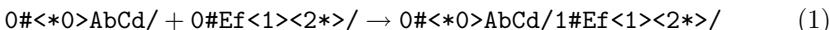


Fig. 2. An example pattern and matching v-molecules

2.3 Recombination Rules

A *recombination rule* is a chemical equation in this AChem. It transforms a group of v-molecules that are matched by the rule's left-hand side patterns into the group of v-molecules expressed by the right-hand side patterns. An example rule is shown below:



When this rule is applied to the v-molecules $0\#WxYzAbCd/$ and $0\#EfGh/$, they are recombined to a v-molecule $0\#WxYzAbCd/3\#EfGh/$.

2.4 Membranes and Cubicles

This AChem has the notion of *membrane*, which models a lipid membrane of living cells. A membrane surrounds a *cubicle*, which models a space surrounded by a lipid membrane, such as cytoplasm. A cubicle has a *reaction pool*, a multiset of v-molecules that can react with each other. A membrane also has a reaction pool; v-molecules in it model protein molecules embedded in the membrane. Cubicles and membranes are called *reaction spaces* in general. A reaction space has its reaction pool and a set of recombination rules. Since membranes can be nested, a *system* of reaction spaces forms a tree (Fig. 3).

A v-molecule embedded in a membrane, or a *membrane v-molecule*, has its direction when it is looked at from one of the neighbouring cubicles. In order to express reactions in which membrane v-molecules are involved, there are patterns to represent such v-molecules. The direction is *top* (indicated by a hat sign (^)) or *bottom* (by an underscore (_)). Figure 4 shows two membrane v-molecules. Both are $0\#AB/$ as v-molecules. If a v-molecule is $\wedge 0\#AB/$ when it is viewed from Cubicle 1, it is $_0\#AB/$ from Cubicle 2 (case (a)). If a v-molecule is $_0\#AB/$ when it is viewed from Cubicle 1, it is then $\wedge 0\#AB/$ from Cubicle 2 (case (b)). So the direction is relative to a cubicle from which the v-molecule is viewed.

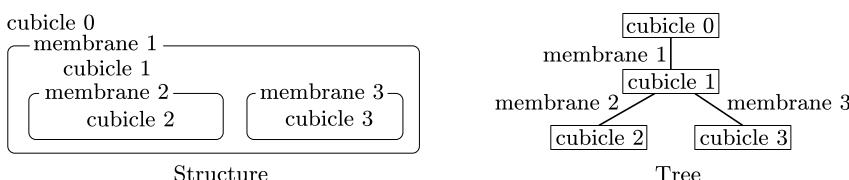
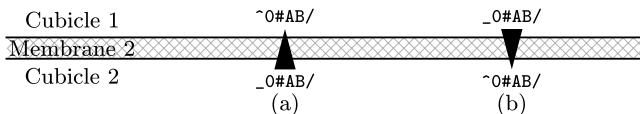


Fig. 3. An example system and the tree structure

**Fig. 4.** Directions of membrane v-molecule

Since membrane v-molecules model membrane protein, reactions between them can also be described. Relative directions of membrane v-molecules are expressed based on the notion of *polarity*: for example, the v-molecules shown in Fig. 4 have the reverse polarities to each other.

Polarity is indicated by an exclamation mark (!) in a pattern. An example rule for reaction between membrane v-molecules is shown below:

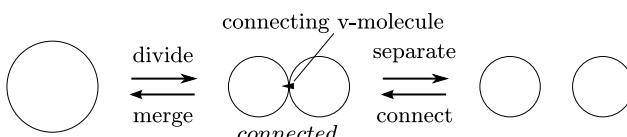


In this rule, $!0\#CD/$ is *negative* to $0\#AB/$ (i.e., $0\#CD/$ has the opposite direction to $0\#AB/$), and $0\#AB/0\#CD/$ on the right-hand side is *positive* to $0\#AB/$. If the first term $0\#AB/$ matches a membrane v-molecule $\sim 0\#AB/$ (from the viewpoint of a neighbouring cubicle), the second term $!0\#CD/$ matches a membrane v-molecule $\sim 0\#CD/$ (but does not match $\sim 0\#CD/$), and the product of the recombination is $\sim 0\#AB/0\#CD/$.

2.5 Connecting Membranes

In the base AChem [6], the division of membrane is modelled as one-step action: one application of recombination rule to a membrane v-molecule may produce two membranes. The extended AChem [5] introduced an intermediate *connected* state (Fig. 5). One membrane is *divided* into two membranes (by applying a recombination rule to one of the original membrane's membrane v-molecules) but they are still *connected* by a *connecting v-molecule*; if a recombination rule to *separate* them is applied to the connecting v-molecule, they are separated. The merger of membranes goes through a connected state as well.

A connecting v-molecule is denoted by an equal sign (=) and membrane identifiers such as $=1(3)0\#AB/$. This v-molecule, of the form $0\#AB/$, connects two membranes whose IDs are 1 and 3. A connecting v-molecule has its direction, as membrane v-molecules do; the parentheses indicate the membrane on the bottom side of the connecting v-molecule.

**Fig. 5.** The dynamics of membranes

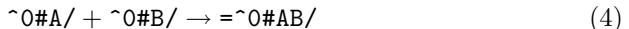
A recombination rule that induces change in the membrane structure includes patterns that represent connecting v-molecules. Such patterns are denoted using a hat sign, an underscore and an exclamation mark in a similar way to denoting membrane v-molecules.

Membrane connection and separation are described by recombination rules. Rule (3) below connects two membranes.



The parentheses indicate the membrane that comes to the bottom side of the produced connecting v-molecule. Swapping the left-hand side and the right-hand side makes a rule for separation.

Division of a membrane is described as the following example rule.



This rule divides a membrane that has membrane v-molecules $\hat{\sim}0\#A/$ and $\hat{\sim}0\#B/$ into two membranes and leaves them connected with a connecting v-molecule of the form $0\#AB/$. The cubicle v-molecules in the original cubicle are distributed to the produced cubicles randomly, and the original membrane v-molecules are also distributed to the two daughter membranes. The new cubicles inherit the recombination rules of the original cubicle, and the new membranes inherit those of the original membrane.

Swapping the left-hand side and the right-hand side of Rule 4 will make a rule for merging. When the reaction spaces are merged, the produced reaction pool is the multiset union of the original two, and the set of recombination rules of the new reaction space is the set union of the original two.

2.6 Dynamics of System

The system is executed on a simulator as follows.

1. Initialise the reaction spaces of the system.
2. Select a reaction space S at random.
3. Select a group of v-molecules in S at random.
4. Recombine them by a randomly-chosen rule R of S , if possible; this may induce structural change in the system.
5. Go to Step 2.

3 Simulating the World of Roguelike Game

In this study, we first construct a system that simulates the world of a roguelike game, then change it into a game. This section illustrates the first part.

A general play of roguelike game is as follows. The human player controls the player character to have it reach the deepest part of the dungeon; the character must overcome obstacles like monsters and traps to accomplish the objective.

In this construction, the player character and monsters are represented as v-molecules, and their behaviour is described by recombination rules. Furthermore, a dungeon, represented as a system of membranes and cubicles, is generated by recombination rules as well.

player character	<table border="1"><tr><td>Brave</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Def</td><td>Atk</td><td>Atk</td></tr></table>	Brave	Nrg	Nrg	Nrg	Nrg	Nrg	Def	Atk	Atk
Brave	Nrg	Nrg	Nrg	Nrg	Nrg	Def	Atk	Atk		
monster	<table border="1"><tr><td>Atk</td><td>Atk</td><td>Def</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Monster</td><td></td><td></td></tr></table>	Atk	Atk	Def	Nrg	Nrg	Nrg	Monster		
Atk	Atk	Def	Nrg	Nrg	Nrg	Monster				

Fig. 6. The v-molecules for a monster and the player character

<table border="1"><tr><td>Brave</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Def</td><td>Atk</td><td>Atk</td></tr></table>	Brave	Nrg	Nrg	Nrg	Nrg	Nrg	Nrg	Def	Atk	Atk
Brave	Nrg	Nrg	Nrg	Nrg	Nrg	Nrg	Def	Atk	Atk	
<table border="1"><tr><td>Atk</td><td>Atk</td><td>Def</td><td>Nrg</td><td>Nrg</td><td>Nrg</td><td>Monster</td><td></td><td></td><td></td></tr></table>	Atk	Atk	Def	Nrg	Nrg	Nrg	Monster			
Atk	Atk	Def	Nrg	Nrg	Nrg	Monster				

Fig. 7. The encounter of the player character and a monster

3.1 Describing Battle

Figure 6 illustrates the forms of v-molecules for a monster and the player character. The v-atoms **Monster** and **Brave** differentiate them. The numbers of v-atoms **Nrg**, **Atk** and **Def** represent the amounts of energy (“hit points”), attacking strength and defence strength, respectively.

The encounter of the player character and a monster is described by (5), which forms a compound v-molecule shown in Fig. 7.

$$0\#<*0>\text{DefAtk}<1*>/ + 0\#<*2>\text{AtkDef}<3*>/ \rightarrow \\ 0\#<*0>\text{DefAtk}<1*>/0\#<*2>\text{AtkDef}<3*>/ \quad (5)$$

To this v-molecule, one of (6) and (7) can be applied.

$$0\#<*0>\text{Atk}<1*>/ - 1\#<*2>\text{NrgNrg}<3*>/ \rightarrow \\ 0\#<*0>\text{Atk}<1*>/ + 0\#<*2>\text{Nrg}<3*>/ + 0\#\text{Nrg}/ \quad (6)$$

$$0\#<*0>\text{NrgNrg}<1*>/1\#<*2>\text{Atk}<3*>/ \rightarrow \\ 0\#<*0>\text{Nrg}<1*>/ + 0\#<*2>\text{Atk}<3*>/ + 0\#\text{Nrg}/ \quad (7)$$

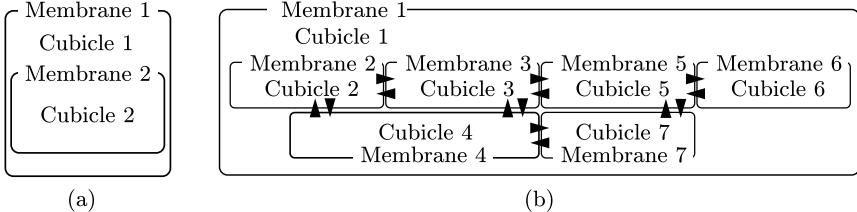
Rule (6) expresses the monster’s damage, and when it is applied, one of the monster’s v-atoms **Nrg** below the player’s **Atk** (i.e., not defended by the monster’s **Def**; the one hatched in Fig. 7) is removed. Rule (7) expresses the player character’s damage in a similar way.

Shown in Fig. 8 are the v-molecules that represent the dead bodies of the player character and a monster. Note the positions of **Brave** and **Monster**; these arrangements prevent (5) from being applied to these v-molecules, so they are inactive.

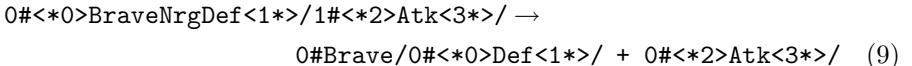
A monster becomes dead when it loses all its energy. This transformation is carried out by (8).

$$0\#<*0>\text{Atk}<1*>/ - 1\#<*2>\text{DefNrgMonster}<3*>/ \rightarrow \\ 0\#<*0>\text{Atk}<1*>/ + 0\#<*2>\text{Def}<3*>/0\#\text{Monster}/ \quad (8)$$

Brave				Atk	Atk	Def
Def	Atk	Atk	Atk			Monster

Fig. 8. The dead bodies**Fig. 9.** The initial state of the dungeon and an example dungeon generated

The player becomes dead in a similar way, by the following rule.

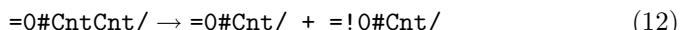


3.2 Generating a Dungeon

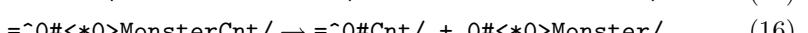
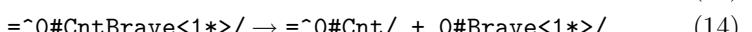
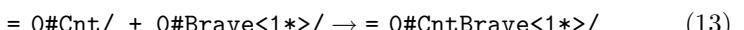
A dungeon in a roguelike game is typically composed of rooms (surrounded by walls) and corridors. We model them as cubicles (surrounded by membranes) and connecting v-molecules, respectively, and define rules to generate a dungeon. Figure 9(a) is the initial state of the system.

Membrane 2 has a number of v-molecules of the form $0\#\text{Cnt}/$; these are raw materials of corridors in the dungeon.

To this system, the following rules are applied repeatedly. They divide a room (10) and connect the produced rooms (11) with bidirectional corridors (12); the process generates a dungeon like the one shown in Fig. 9(b). The structure of dungeon continues to change until all the membrane v-molecules of the form $0\#\text{Cnt}/$ transform to connecting v-molecules $=0\#\text{Cnt}/$.

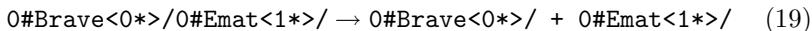
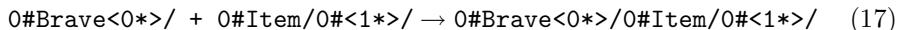


The following rules enable the player character and monsters to go through corridors.



3.3 Picking Up and Using Items

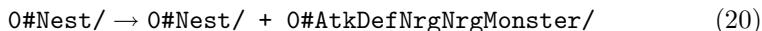
In a usual roguelike game, the player character picks up and uses various items. In this game, an item is represented by a v-molecule. The following rules enable the player character to use a bomb.



A v-molecule $0\#\text{Item}/0\#\text{Emat}\text{EmatBomb}/$ represents a bomb, and (17) attaches it to the player character (which means the character picks it up) to make a compound v-molecule $0\#\text{BraveNrgNrgDefAtk}/0\#\text{Item}/0\#\text{Emat}\text{EmatBomb}/$. The player character uses the bomb by (18), and then the bomb is released by (19). When it hits a monster, the monster is damaged.

3.4 Creating a Monster

In a usual roguelike game, monsters come out of nowhere. The following rule expresses it.



4 Changing Simulation into a Game

The previous section described the system to simulate the world of the roguelike game, in which the player character and monsters move autonomously, requiring no human intervention. Now we change the simulation into a game, in which the human player controls the use of items obtained by the player character.

To do this, we remove a rule. The player character picks up an item by (17) and uses it by (18), and the item works by (19). So we remove (18) from the system, and add code outside the AChem to control the use of items. Figure 10 shows a snapshot of the obtained game.

```

xterm
The brave entered the dungeon.
>
Found an item,
>g
Put "bomb" in the bag
>u

Items:
  name: quantity
  bomb: 1
  potion: 3

which item will you use?;bomb
Leave "bomb" in the room,
>

```

Fig. 10. A snapshot of the developed game

5 Discussion

A significant result of the study is the brevity of description that defines the simulation. A set of 22 rules describes the generation of dungeon, battle between the player character and monsters, the movement of them in the dungeon, the player character's picking up and using items (bombs and healing potions), and creating monsters. When this type of game is implemented with a traditional programming language, the programmer must use conditional statements and iterations. But the AChem implies them: v-molecules that are matched by the left-hand side of a rule are selected and transformed, and this process is iterated. The whole process is automatically performed by the simulator. Moreover, the AChem simulator uses random numbers to choose v-molecules and rules. It may eliminate coding for, say, selecting actions using random numbers, which is typically required in traditional game programming. For example, in our game, choosing a monster to meet the player character is not explicitly described using a random number; neither is deciding which of the player character and the monster making a battle v-molecule wins. These properties seem to be beneficial not only to implementing a roguelike game but also another type of game if an element of the game meets another element probabilistically and the meeting gives rise to a certain event, like chemical reaction.

Using the AChem, we took a specific approach to make the game: first, we built a system to simulate the world of the game; then we removed a rule to change the mere simulation into a game. Specifically, we removed the rule for using items and left the function outside the AChem to have it controlled by the human player. This method — to design simulation and then to remove a function — seems a general way to design a game based on simulation. Suppose we remove a rule to create monsters, instead of the one for using items, from the simulation and let the human player control the creation. Then a different game is obtained, in which the human player thwarts the hero's adventure. A famous simulation game, SimCity, can be viewed in a similar way. The player plays the role of government of the city to design the city. This game simulates everything about the city but the government. Panic Shooter [2] is a shooter game built on cellular automata. The behaviour of bullets and barriers is defined as rules of cellular automata. The player proceeds toward a designated goal, shooting or avoiding barriers. This is similar to our approach, in which the world of game is defined by rules for simulation, and the only part the player controls is outside the simulation.

6 Concluding Remarks

In this paper, we have illustrated the development of the roguelike game in the artificial chemistry with membranes. We took the two steps to develop it: first, we built an AChem system to simulate the world of the game; then, we removed a rule for using items and left the function outside the AChem simulator. We speculate this “simulation-minus-one” approach may be useful to develop simulation games. Since AChems are tools for simulation, they may be an effective

platform for building such kinds of games. We also showed how easy it is to do “minus one” in the AChem.

Because the “minus-one” part is not dealt with by the AChem simulator, its function must be implemented by an outside mechanism, such as a program written in a traditional programming language. A framework to support this implementation will enhance the effectiveness of the present approach.

References

1. Electronic Arts Inc.: The official Spore and Spore creature creator site, <http://www.spore.com/ftl>
2. Tosik: Cell automaton wo motiita shooting game (in Japanese), <http://d.hatena.ne.jp/tosik/20070409/1176081337>
3. Kim, K.J., Cho, S.B.: A comprehensive overview of the applications of artificial life. *Artificial Life* 12(1), 153–182 (2006)
4. Dittrich, P., Ziegler, J., Banzhaf, W.: Artificial chemistries – a review. *Artificial Life* 7(3), 225–275 (2001)
5. Watanabe, T.: Extending an artificial chemistry by incorporating membranes. Master’s thesis, School of Computer Science, Tokyo University of Technology, Tokyo, Japan (2008) (in Japanese)
6. Tominaga, K., Watanabe, T., Suzuki, M.: Formulating membrane dynamics with the reaction of surface objects. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS, vol. 4648, pp. 12–21. Springer, Heidelberg (2007)

Evolving Simple Art-Based Games

Simon Colton and Cameron Browne

Computational Creativity Group
Department of Computing, Imperial College London
{sgc,camb}@doc.ic.ac.uk
<http://www.doc.ic.ac.uk/ccg>

Abstract. Evolutionary art has a long and distinguished history, and genetic programming is one of only a handful of AI techniques which is used in graphic design and the visual arts. A recent trend in so-called ‘new media’ art is to design online pieces which are dynamic and have an element of interaction and sometimes simple game-playing aspects. This defines the challenge addressed here: to automatically evolve dynamic, interactive art pieces with game elements. We do this by extending the Avera user-driven evolutionary art system to produce programs which generate spirograph-style images by repeatedly placing, scaling, rotating and colouring geometric objects such as squares and circles. Such images are produced in an inherently causal way which provides the dynamic element to the pieces. We further extend the system to produce programs which react to mouse clicks, and to evolve sequential patterns of clicks for the user to uncover. We wrap the programs in a simple front end which provides the user with feedback on how close they are to uncovering the pattern, adding a lightweight game-playing element to the pieces. The evolved interactive artworks are a preliminary step in the creation of more sophisticated multimedia pieces.

1 Introduction

Broadly speaking, evolutionary art is the process by which programs are evolved which are able to produce artworks. In most cases, the user acts as the fitness function, by selecting and rejecting genotypes (programs) based on their phenotypes (the images they produce). Example projects include Mutator from Latham et. al., where sequences of 3D shape morphing operations are evolved [7], NeVar from Machado et. al., where programs are evolved to choose each pixel’s colour in an image based on its location [4], and the work of Jon McCormack, where L-systems for 3D-modelling of organic forms are evolved [5]. In none of these projects are the artworks generated interactive in any way. In contrast, a major trend in so-called ‘new media’ approaches to modern visual arts is to build online, dynamic and interactive artworks, often with a puzzle or game element to them. The question we therefore address here is whether it is possible to automatically evolve such artworks, i.e., of a dynamic and interactive nature and with a lightweight game element to them.

To answer this question positively, we extend the Avera evolutionary art system (which is described below) by both extending the sophistication of the

programs it evolves and by building a simple front-end to use the programs in an interactive way. As described in section 2, we simulate a well known form of making artworks, namely spirographs, by evolving programs which control the position, rotation, size, colour and transparency of a series of geometric shapes (namely squares, circles and triangles). As described in section 3, we extend this to enable the generation of programs that produce spirographs drawn in a way dependent on where a user clicks in the canvas that the spirograph is being drawn on. In section 4, we describe how this enabled us to add a puzzle element to the pieces, by evolving programs which embed a causal pattern of clicks to be discovered by the user. If the user correctly approximates the timing and position of the required clicks, they are rewarded by generating an aesthetically pleasing spirograph, and we provide a target image to guide his/her clicks. We conclude by describing some of the ways in which we plan to extend this work to produce more sophisticated multi-media art/game pieces.

1.1 Background

Our interactive system is an extension of an earlier framework designed for creative artefact generation, named Avera [3]. This framework proved general enough to accept problems from a range of different domains, yet concise enough to allow for quick prototyping and easy modification. It is based on a variable-sized tree representation to encode solutions, which admits an evolutionary search through crossover and mutation. The framework uses a combination of node-type constraints upon the generated parse trees, and logical constraints for removing unwanted node patterns. The former is based upon the constraints of Montana's strongly typed genetic programming [6], and allows the type systems of programming languages to be respected. Meanwhile, the latter allows for constraints over node dependencies to be expressed. Finally, a method is provided for translating the tree structure used internally to represent solutions into text output, which is analogous to the genotype-phenotype mapping in Genetic Programming. This can be used to convert solutions into entire programs, program classes (in particular, compilable Java classes), scripts or data that can be accepted by other programs, and the behaviour of these programs can be used to evaluate the success of the solution. An example tree and the resulting code are given in the next section.

By attaching a front-end which enables the user to act as a fitness function by choosing between phenotype images, in [3], the authors applied Avera to three evolutionary art projects involving: 1) pixel-based, 2) image filtering and 3) particle-based styles. We now describe an additional mode of operation for interactive use.

2 Evolutionary Setup for Spirographs

The word ‘spirograph’ is commonly used to describe both an art-producing contraption – most commonly used as a children’s toy – and the artworks that

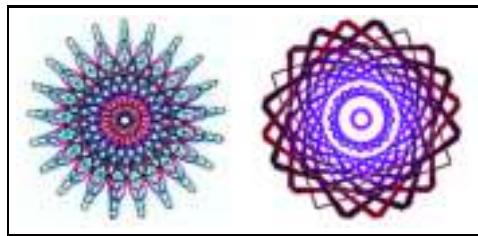


Fig. 1. A hand-drawn spirograph curve from en.wikipedia.org/wiki/Spirograph, and an approximate spirograph produced using the repeated placement of squares

it produces.¹ Introduced in the 1960s, the contraption is able to produce hypotrochoid and epitrochoid curves. While these curves are continuous, the look of spirographs can be approximated by repeatedly drawing geometric objects, including squares, circles and triangles in close proximity to each other. For instance, Figure 1 portrays a spirograph produced by hand using the contraption, and an image consisting of a series of coloured squares. To evolve programs able to generate such approximate spirographs, we used Avera to construct a Java class with the following two methods:

- **initialise**: In this method, the type of shape (square, circle, triangle) is specified, along with an upper limit for a time-step counter. These values do not change as the image is drawn.
- **update**: In this method, the position of the centre, and the scale, rotation, transparency and RGB colour values of a shape are calculated. These values may be constant, or may be a function of the time-step counter and/or the image square size (which we kept at 500 by 500 pixels in all the examples reported here, but which can be varied).

Starting at zero, as the time-step counter increases up to the maximum specified in **initialise**, the shapes change according to the functions in **update**, and if the shapes are drawn onto a canvas, the resulting image approximates a spirograph image. We used the integers between 1 and 20 as terminals, along with a pointer to the time-step counter and the size of the canvas. We also used the binary functions $\cos(x)$ and $\sin(x)$ as well as addition, multiplication, division and subtraction, and the unary functions: $random(x)$ (able to generate a random number between 1 and x inclusive) and negation. Note that the absolute value of the output of all unary and binary functions was taken, as all the values which define a shape (with the exception of rotation) need to be positive integers. In addition, before drawing the shapes, any RGB values that exceeded 255 were limited to 255. Moreover, for the time-step constant in **initialise**, we take the constant that nt is equated to, and use $200 + (100 * nt)$ time steps.

In Figure 2, we present an example of an evolved spirograph-generating program. In this example, the **initialise** program (not shown) specifies a square

¹ For an online applet which creates spirographs, see the following web site:
<http://wordsmith.org/anu/java/spirograph.html>

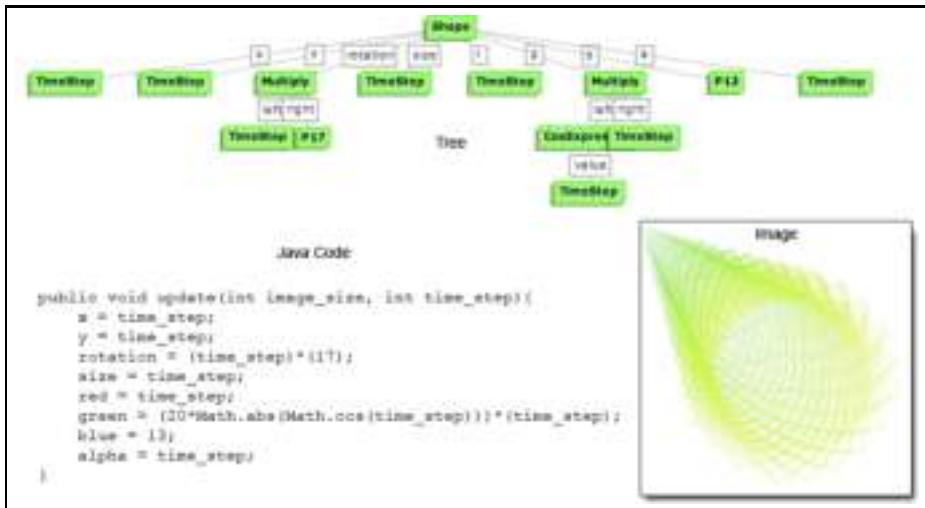


Fig. 2. An evolved tree, its java interpretation and the resulting image

as the shape to draw, and 250 time steps. As we see, in the `update` method, the x and y co-ordinates are equated to the time step, and hence the square moves from the top left to the bottom right of the canvas. The size of the square increases in line with the time step, as does the rotation and the transparency of the line drawn. The blue value of the square is fixed, but the red value increases with the time step, and the green value is calculated as: $20 * |\cos(\text{time_step})| * \text{time_step}$.

3 Adding Dynamic and Interactive Features

Drawing shapes onto a canvas can be achieved very quickly, but if the process is slowed down – we used a 40ms delay between the placing of subsequent shapes – then the static spirographs essentially become a dynamic, viewable video. The Avera specification was then extended so that the programs could use the (x, y) co-ordinates of a mouse click in the functions which determine the colour, position, rotation, size and transparency of the shapes. A simple graphical user interface was built to dynamically display the spirograph and to process mouse clicks by passing them to the drawing process, so that the evolved `update` method can use the click point to change the shapes as they are drawn, which adds an interactive element to the pieces. The click point defaults to the middle of the canvas (250, 250) until the user clicks the canvas.

As an example, the `update` program embedded in the Java class producing all four images in Figure 5 is given in figure 3. We see that the x and y values are fixed to the centre of the canvas, the rotation of the shape increases with time, the size is proportional to how far down the canvas the user clicks, and the colour is dependent on whereabouts from right to left the user clicks. In practice, without any clicks, this spirograph rotates a blue square with RGB value (10, 10, 250) and of size 250 around the centre point of the canvas. However, if the user clicks on a

```

public void update(int image_size, int time_step, Point click_point){
    x = image_size/2f;
    y = image_size/2f;
    rotation = time_step;
    size = click_point.y;
    red = 10;
    green = 10;
    blue = click_point.x;
    alpha = 255;
}

```

Fig. 3. Evolved update function for the images portrayed in Figure 5

point on the left of the canvas, the colour becomes blacker, going blue again if the user clicks on the right of the canvas. If the user clicks towards the top of the canvas (noting that y co-ordinates *increase* from top to bottom), the square gets smaller, but reaches a size of 500 as the user clicks towards the bottom of the canvas. Note that – for reasons explained later – in the first, third and fourth spirographs of Figure 5, the user clicked along the centre line from the top to the bottom of the canvas. A constraint dictating that the evolved methods must contain some reference to the mouse click point was added to the Avera specification.

4 Adding Game Aspects

A game need not have explicitly stated goals to be enjoyable, according to the recent concept of *ludic engagement* proposed by Gaver and et al [1]. Ludic engagement covers a wide range of exploratory, self-motivated activities with which participants engage not to achieve some particular goal but because the process itself is rewarding. The resulting interactions are led by pleasure rather than utility, and can be valuable in helping to explore new relations and to foster an awareness of perspectives. The kinds of art pieces we evolved are meant to be enjoyable to interact with in their own rights. However, we added a game element to further demonstrate the possibilities for our approach, and to hopefully make the pieces more interesting by encouraging the user to interact with the programs for a purpose.

4.1 Introducing a Game Element

To instil a game element, we added to the programs that Avera evolved a notion of a fixed causal/spatial sequence of mouse click points. When viewing the phenotypes in Avera's browser (i.e., when we act as the fitness function in order to evolve spirograph programs), we display them as they would look if the user were to click exactly the specified locations on the canvas at exactly the specified times. This enabled us to evolve spirograph programs which can produce aesthetically pleasing images if the user is able to reproduce the correct set of on-screen clicks. By hiding the click sequence, but giving the user a target image to try to produce through interaction with the spirograph generator, we therefore gave the programs a lightweight game element.

We found that having to uncover an arbitrary set of points at arbitrary times was too difficult, so we opted for fixed sets of fairly natural patterns which dictate the placement of the points, and a fixed set of Morse-code style time intervals which dictate the intervals (in terms of the time steps of the spirograph generator, not actual seconds) that the user should leave between clicks. To enable this, we added two variables to the `initialise` methods that Avera evolved, and stipulated that the values correspond to position/interval sequences respectively. We extended the GUI described above by enabling it to read the evolved Java classes, extract the pair of values representing the sequences and calculate the set of triples $\langle x, y, t \rangle$ which specify that the screen should be clicked at co-ordinate (x, y) at time-step counter t . Then, as the shapes are drawn onto the canvas, the clicking of points are simulated automatically, triggered by the time-step counter reaching certain values.

The position patterns and time interval sequences we implemented for these preliminary tests were as follows:

- *Position patterns:*

corners (4 points), clockwise circle (8 points), anti-clockwise circle (8 points), top-bottom centre line (6 points), left-right centre line (6 points), plus sign (8 points), cross sign (8 points), left-right diagonal from top to bottom (5 points), left-right diagonal from bottom to top (5 points).

- *Time interval sequences:*

$\{10\}$, $\{10, 20\}$, $\{10, 40\}$, $\{10, 20, 30\}$, $\{10, 20, 40\}$, $\{10, 20, 10, 30\}$.

As an example, suppose the shape class evolved by Avera specifies that the position pattern variable is set to 2 (signifying an anti-clockwise circle position pattern), and the time-interval variable is set to 2 also (signifying a dot-dash time interval pattern, using the Morse-code analogy). The time intervals signify the length of time the user needs to wait after they have clicked the screen, before clicking the screen again. In Figure 4, we portray the sequence of clicks the user would need to perform in order to recreate the target image.

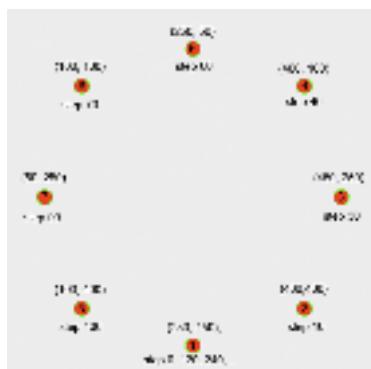


Fig. 4. An example of required click positions in order to recreate a target spirograph

4.2 Improving Enjoyability

With 9 position and 6 time-interval sequences, there are 54 possible click sequences available for Avera to embed into the Java classes that it evolves, which is enough for trial and error to become an unfavourable option. Hence the user will need to experiment to begin to understand how their clicks affect the drawing of the shapes, and then decide through experimentation which sequence of clicks produce the desired spirograph. We kept the first iteration of the GUI very simple. In addition to the panel allowing the user to click the changing spirograph, the GUI also contained an image of the completed target spirograph, and the user's first click started the time-step counter. They were then required to keep on clicking until the counter reached its maximum, or start again using a button we provided. Unfortunately, we found this GUI far too difficult and haptically awkward to recreate the target image. Hence, in a second GUI iteration, we implemented the following improvements:

1. The user is allowed to drag the mouse, and the `update` function driving the spirograph generation is given the mouse drag point just like it is given mouse click points. We found that this encouraged the user to play around with the program much more at the start of a session, when trying to uncover exactly how positions on screen corresponded to shape changes. This was because dragging is essentially equivalent to hundreds of click operations.
2. Once the user has determined what they hope is the correct sequence of clicks, they can restart the program, click the pattern, and end where they started with a right-click operation. This then informs the GUI to repeat the same pattern of clicks until the drawing process ends. This removed the requirement for the user to click until the end of the process, which really limited their enjoyment.
3. We added a button which enabled the user to see a simulation of the correct click sequence, so that they could be given help in recreating the target image.
4. With many examples, where the `update` method used a complicated function involving the click co-ordinates to determine the shape change, it was difficult to comprehend how a click altered the shapes. Hence, we implemented a scoring mechanism which gave the users instant feedback on how close their sequence of clicks has been to the correct sequence. This is calculated as an average over the distance between a user's click and the relevant required click. The required click which is closest to, but causally before, the user's click is taken as the relevant click. At the start of the game, the user can speculatively click around the screen in order to determine where the starting point of the pattern is, and the scoring mechanism gives valuable feedback towards a solution.
5. Even when the user has correctly determined the pattern to click and the correct times at which to click (or they have looked at the solution), it was still impossible to recreate the target image in many cases. This is because some of the spirograph programs were particularly sensitive to slightly wrong clicks (both spatially and causally). This was very dissatisfying for users, as they felt they should be rewarded for their endeavours with a recreated target

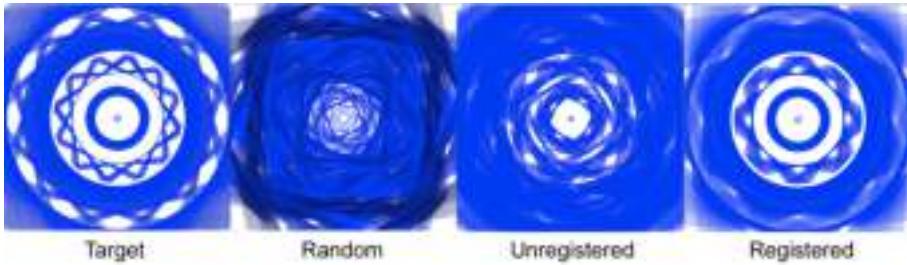


Fig. 5. Target spirograph; spirograph produced via random clicking; spirograph produced by near-correct clicking, but without point registration; spirograph produced by near-correct clicking, with point registration

image. To solve this problem, we perform point *registration*. That is, when the user clicks on a point which is within 20 time steps (either before or after), and within 50 pixels of a correct click, it is mapped to that correct click. Hence, when the user finishes the sequence with a right-click, if they have been nearly right with all of their clicks, the resulting image produced will be almost identical to the target image (with any differences arising from the slightly incorrect early generation of shapes before the repetition was turned on). Figure 5 shows the value of this addition to the GUI. In this example, the correct sequence is the centre line from top to bottom of the canvas, with repeating intervals of 10, then 20 milliseconds. Comparing the image produced by a random set of clicks (image 2 in Figure 5) with the target highlights the fact that some thought is needed to solve the problem. However, in the final two spirographs of Figure 5, the user had uncovered the correct sequence of clicks. In the first case, registration was not enabled, and the resulting spirograph is far from the target image. In the second case, however, registration was enabled, but one of the clicks was not close enough to be mapped to the perfect click (hence it is not a perfect reproduction of the target image, but it does show the value of registration, as the resulting spirograph is visually much closer to the target one than the third image).

6. Finally, we added a slider bar, so that the user can alter the number of milliseconds delay between shape renders, ranging from 0 to 100 milliseconds. This enables them to work slower or faster depending on their abilities and/or the difficulty of the puzzle.

5 Conclusions and Future Work

We have so far performed only limited testing of the interactive and game-playing aspects of the evolved pieces, which prompted the second round of GUI implementations. We plan to perform much more user testing to see if the particular configuration we have is indeed enjoyable, and to see where further improvements can be made: for instance, we hypothesise that mouse drag gestures will provide a more natural input to the pieces than clicks. To do the user testing, we will appeal to internet communities to provide feedback about the pieces, which

we will make available as online applets. While the games may be simplistic (albeit quite difficult in some cases), the visual variety of the artworks underlying them is quite marked. To emphasise this, during a single session lasting around 2 hours, we evolved 3000 spirograph programs in 30 generations of 100 individuals. We selected around 150 as our favourites (functionality which Avera enables), and we present 77 of these in the appendix. There is considerable variety in the results; due to the ability to use random numbers, some of the images have a particularly sketchy look, while others include vibrant block colours resembling abstract art, and others still have an unexpected photo-realistic quality.

We hope to have shown that there is much potential for evolutionary search to break through from static visual art into the more dynamic, interactive, puzzle-based world of new media art. We plan to further expand the sophistication of the programs we evolve so that their phenotypes are multimedia art pieces on a par with those written by hundreds of new media artists using packages such as processing [2]. By further adding the ability for Avera to use a fitness function automatically (i.e., bypassing the requirement for a user's inputs, as can be the case with the NeVaR system [4]), we hope to show that computational creativity approaches can be used to generate increasingly complex, valuable and enjoyable artefacts on the border of the art and games worlds.

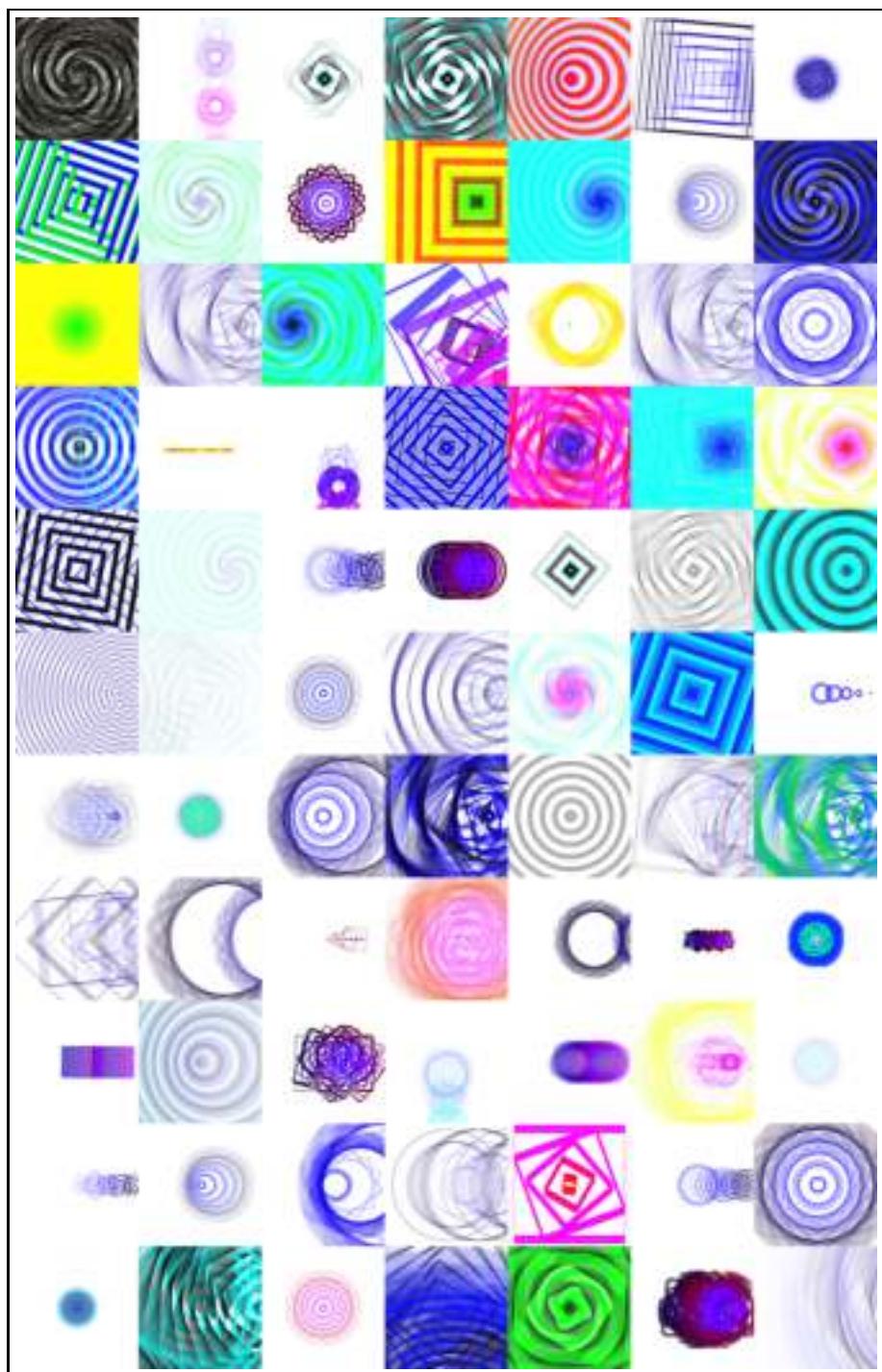
Acknowledgements

This work was supported by EPSRC grant EP/E005713. Thanks to Jeremy Gow for useful input, the anonymous reviewers for helpful suggestions, and to Marc Hull for developing the original Avera system.

References

1. Gaver, W., Bowers, J., Boucher, A., Law, A., Pennington, S., Walker, B.: Electronic furniture for the curious home: Assessing ludic designs in the field. *International Journal of Human-Computer Interaction* 22(1/2), 119–152 (2007)
2. Greenberg, I.: Processing: Creative Coding and Computational Art (Foundation). Friends of Ed. (2007)
3. Hull, M., Colton, S.: Towards a general framework for program generation in creative domains. In: *Proceedings of the 4th International Joint Workshop on Computational Creativity* (2007)
4. Machado, P., Cardoso, A.: NEvAr – the assessment of an evolutionary art tool. In: *Proceedings of the AISB 2000 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science* (2000)
5. McCormack, J.: Impossible Nature. *The Art of Jon McCormack*. Australian Centre for the Moving Image (2004)
6. Montana, D.: Strongly typed genetic programming. *Journal of Evolutionary Computation* 3, 199–230 (1995)
7. Todd, S., Latham, W.: *Evolutionary Art and Computers*. Academic Press, London (1992)

Appendix: 77 Examples Evolved in a Single Session



Swarming for Games: Immersion in Complex Systems

Sebastian von Mammen¹ and Christian Jacob^{1,2}

¹ University of Calgary, Dept. of Computer Science, Calgary, Canada

² University of Calgary, Dept. of Biochemistry & Molecular Biology, Calgary, Canada

Abstract. The swarm metaphor stands for dynamic, complex interaction networks with the possibility of emergent phenomena. In this work, we present two games that challenge the video player with the task to indirectly guide a complex swarm system. First, the player takes control of one swarm individual to herd the remainder of the flock. Second, the player changes the interaction parameters that determine the emergent flight formations, and thereby the flock’s success in the game. Additionally, a user-friendly interface for evolutionary computation is embedded to support the player’s search for well-performing swarm configurations.

1 Introduction

AI algorithms have been successfully applied in computer games for pattern detection in human-computer interfaces [5] or for non-player character optimization, i.e. shortest path search [10] and planning [11]. We propose that bio-inspired methodologies might directly offer new approaches to game principles.

In this work, artificial swarms confront the player with a novel perspective: (1) When partaking in a flocking collective, and (2) when conjuring and utilizing emergent flocking formations by regulating a swarm’s underlying interaction parameters. We present game concepts including mission statements, level design and user-interaction elements that support both approaches. The interaction patterns in artificial swarms can be highly dynamic and complex. In order to handle this complexity, we provide evolution-based mechanisms to breed swarm configurations in an intuitive manner. To our knowledge, no similar swarm-based games have been presented in the past.

In the next section, we briefly introduce the scientific background for the presented work and related games. In the two subsequent sections we draw a comprehensive picture of two fully-featured, swarm-based games. Finally, our results are briefly summarized, and we suggest future steps for expanding our swarm-based gaming concepts.

2 Swarms and Emergent Gaming

In 1987 Reynolds presented a simulation concept that mirrors the flocking dynamics of birds [12]. In his model, a simulated virtual bird i , also called *boid*, is

represented as a pyramidal graphical object oriented towards its velocity \vec{v}_i . In order to be able to react to its neighbors, a boid possesses a conic field of perception that is determined by a radius r and an angle α . Rather loosely, Reynolds explained which accelerating urges are triggered by the flocking mates perceived in their neighborhood. *Alignment* adjusts the direction to the neighbors, *cohesion* draws the individual towards its neighbors, and an urge for *separation* prevents the boids from bumping into each other. Combined with some randomness, these rudimentary urges suffice to let large crowds of agents flock smoothly in virtual 3D spaces.

Many AI development frameworks provide a basic boid implementation [4, 8, 1, 18]. Our present boid simulation mainly follows the implementation discussed in [9, 16]. In a first step, the neighborhood of each individual is re-calculated based on position updates of the last iteration. The set of neighbors is used to calculate the urges of separation, alignment, and cohesion, as well as an urge towards the world center and a random vector. The classic boid urges are normalized through division by the number of considered neighbors, the world center and the random urges renormalized to unit-vectors. An individual's acceleration is computed by the weighted sum of urges, whereas the weights are part of the swarm agent's configuration. Both acceleration and velocity are limited by a maximal value (again part of the configuration). In a second step, each boid's location is (simultaneously) updated.

As repulsion from the neighboring mates can lead to sudden flock dispersions, we calculate the separation urge as the sum of deflection vectors from the mates. The deflection is calculated by reflecting the vector from the individual to its mate in respect to the individual's velocity. Alignment is simply implemented as the sum of the neighbors' velocities. Cohesion is calculated as the sum of distance vectors. Detailed formulas are provided, for instance, in [16].

Knowing the swarm individuals' positions and orientations, as well as the dimensions of their fields of perception, the interdependencies of a given swarm can be inferred. However, we implemented and tested two visualization methods in order to enhance the gaming experience. Figure 1(a) shows a set of boids flocking in a loose cluster formation. The individuals' orientation is indicated by their rotation. In Figure 1(b) the fields of perception are illustrated. The visualization methods in Fig. 1(a) and 1(b) are adapted from Reynolds' original boids publication [12]. When connecting all neighboring boid agents as in Fig. 1(c), the flock appears as a continuously changing interaction network, as suggested in [16]. The field of view visualization (Fig. 1(b)) allows the player to better understand the swarm's movements. The network visualization (Fig. 1(c)) can facilitate the game play, especially since dependency chains are depicted that determine the success of herding or dragging tasks in the game, as we explain in Section 3.

The idea to use artificial swarms in video games becomes obvious when experiencing interactive swarm art installations. Jacob et al., for instance, have demonstrated the interactive potential of swarm simulations steered by video-captures of the audience's movements [6]. The natural feel to its smooth motions,

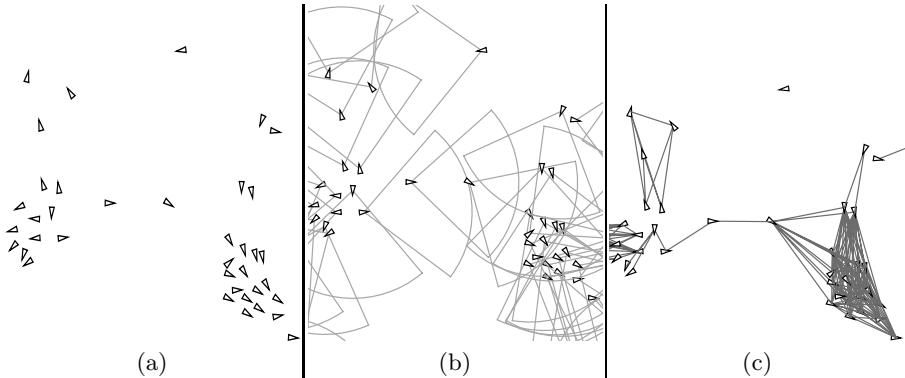


Fig. 1. (a) A boid flock in a loose formation. (b) The geometric fields of perception are illustrated. (c) Edges depict the qualitative interaction network.

the perceived liveliness of the swarm motivates the audience to explore its responsiveness and to enjoy swarm-generated melodies or virtual paintings drawn during the flock's movements. The interactivity of artificial music-generating swarms even supports live performing musicians [3, 2]. An example for an autonomous model is given in *AtomSwarms* where music generating agents evolve and self-organize in a complex eco-system [7]. Within *AtomSwarms* and during live performances a conductor can change the population size of the interactions, but is not provided with the means to directly interfere in the simulation. In other contexts, artificial swarms were bred in well-directed evolutionary runs. The characterization of boid flocks based on the average neighborhood perception allowed to track the changes within the boid interaction network and breed, for instance, an oscillating flock formation [16]. On the other hand, boid swarms were bred by means of interactive evolution to exhibit various flocking formations [9]. Interactive evolution was also applied to explore the capabilities of swarm systems called *Swarm Grammars* that build three-dimensional structures [15]. In these experiments, so-called *breeder volumes* were introduced that apply the genetic operators mutation and recombination to swarm individuals that are passing through.

3 Herding Cattle

In our first swarm-based game prototype, the player is directly immersed in the process of flocking formation. In particular, the player's task is to influence the overall flock, while only navigating one swarm individual. In order to manage this task, one quickly has to gain an intuitive idea of how to control an individual's flight pattern, while taking into account neighborhood relations and affects on overall acceleration behaviors. These skills, in tandem with the player's fast identification of, for example, appearing obstacles and reaction abilities are the ingredients of this basic game concept.

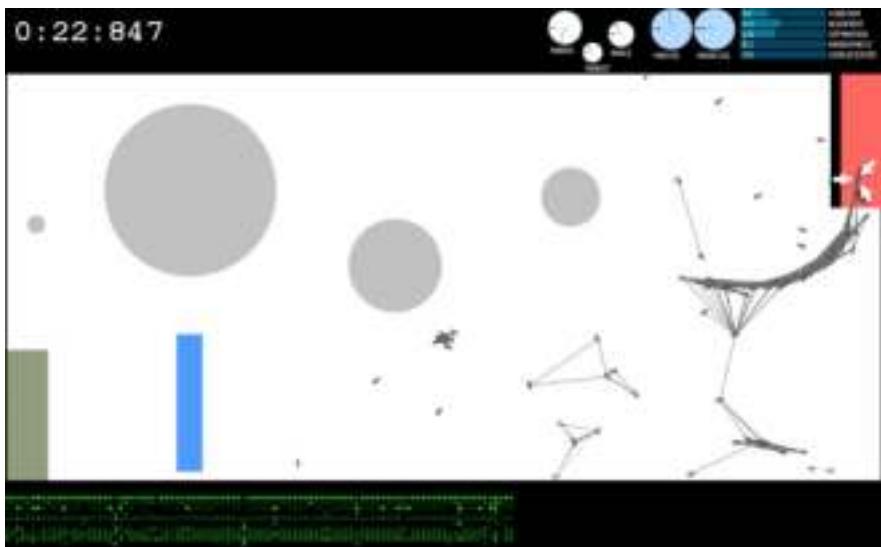


Fig. 2. Screenshot of a 2D boid herding game. The player's swarm individual (highlighted by three white arrows) drags a "tail" of associated agents into the goal.

A simple game setup challenges the player to get a number of swarm agents from a starting area to a goal location. Figure 2 shows a screenshot of an according implementation¹ in the programming environment Processing [1]. In the following paragraphs, the various elements of the screenshot are magnified and explained in detail.

3.1 Playground Interactions

At the beginning of the game, the swarm individuals are positioned in the bottom left corner of the playground (Fig. 3(a)). The player has to herd the flock into the upper right corner (Fig. 3(b)) only by steering one individual that follows the coordinates of the mouse. When the player steers its individual beyond the perceptual range of its peers, they fall behind. If the guiding individual moves too swiftly, it does not exercise a great impact on its mates' flight momentum. On the way to the goal, there are several obstacles denoted as circles and rectangles. The swarm agents deflect from the four circles (Fig. 3(c)), bounce off the dark bar next to the goal ((Fig. 3(d))), and change their flocking configurations, if they pass through the stripe below the biggest circle. The latter effect is, in fact, implemented as a genetic mutation operator [15]. Analogously, the swarm agents' configurations can be understood as their genotypes, their geometric representation and exhibited behaviors as phenotypes.

¹ Its development started on the basis of Daniel Shiffman's boid implementation "Flocking" that is included in Processing's examples library.

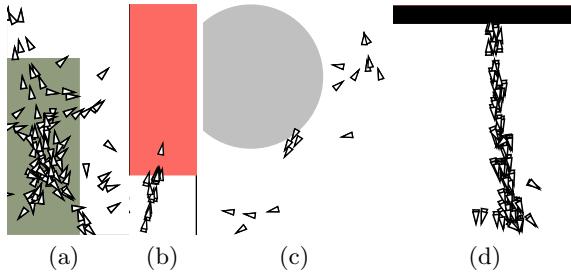


Fig. 3. Level design constituents of a herding game prototype

3.2 GUI for Genetic Manipulation

Different information about the game is displayed on the black stripes at the bottom and the top. In the upper left corner, the time is shown that has elapsed since the game started. As in many other games, timing is a simple means to measure the efficiency of the player's mastery. On the right-hand side of the upper stripe are three knobs to adjust the swarm agents' fields of perception (Fig. 4(a)) and two more to set the maximal values for acceleration and velocity (Fig. 4(b)). They are followed by five sliders that control the weights of the agents' flocking urges (Fig. 4(c)). The last slider is colored in a different shade. It does not change the swarm's properties, but changes the integration step size of the simulation. Changes through this graphical user interface² are applied either to all swarm agents or to selected individuals (Fig. 4(d)). Differences among the individuals are depicted in the bottom graphic in Figures 2 and 4(d). The parameter values are mapped to color values between 0 and 255 corresponding to black and bright green, respectively. One column of pixels represents the genotype of one individual.

4 Crows in the Cornfield

In a second swarm game, the player does not interact with the swarm spatially, but exclusively changes its genotype. The metaphor of the game is a flock of crows that feeds on corn. The same GUI as in the previous game is utilized to configure and thereby direct the flock towards the corn and to maximize the harvest. At the same time, the swarm individuals interact with various objects (as in Section 3), and are even endangered to fly into deathtraps. The latter ones are depicted as the bright rectangular areas in Figure 5. The even brighter, circular spots represent growing corn, the black areas repelling objects. The game ends when all swarm individuals have died. The objects on the playground along with the crops are constantly scrolling towards a random direction. Thus, the flock is confronted with a changing configuration, and the player has to be quick-witted to ensure the flock's survival and a successful harvest. Two GUI elements are different from the previously described herding game: (1) The harvesting score is displayed in the upper-left corner. (2) An additional slider is depicted

² We rely on the controlP5 GUI library by Andreas Schlegel (<http://www.sojamo.de>).

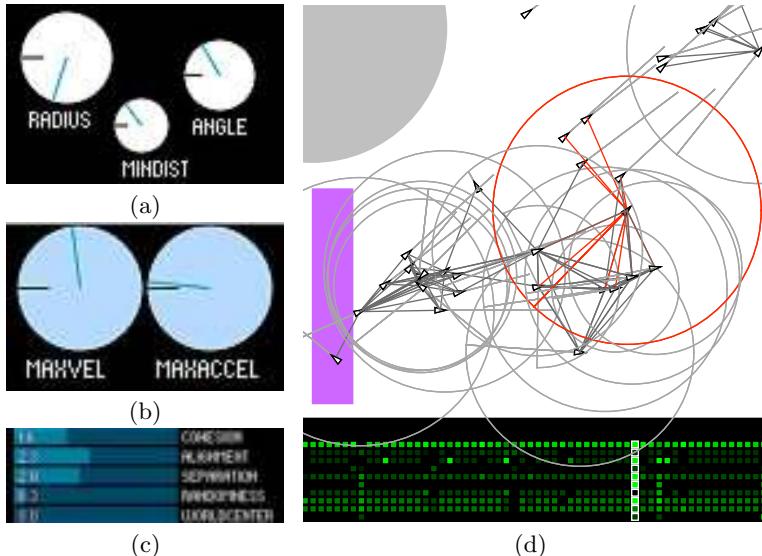


Fig. 4. Graphical user elements allow to change (a) an agent's field of perception, (b) its maximal velocity and acceleration, and (c) its flocking weights. (d) A selected agent is rendered in a different color.

left of the knob for radius adjustments. By means of this slider, the player can choose one of a set of previously stored swarm genotypes. The selected swarm genotype is immediately deployed in the game, thus facilitating the player's task of parameter-based, indirect swarm navigation.

4.1 Evolving Swarm-Macros

As part of the game, but before the player is actually urged to control the flock, two ways are offered to create an assortment of swarm configurations. As seen in Section 3.2, swarm configurations can be manually designed and tested. Alternatively, the player may rely on computational evolution to automatically breed a swarm genotype. By placing a number of tiles on the playground, the player determines where the swarm individuals should be flocking (Figure 6). This alternative evolutionary way to give rise to swarm configurations with specific trajectories is quite important. Manual adjustment of a swarm's flocking parameters can be a very challenging task, that could easily frustrate the player. The implemented breeding approach is similar to the evolution of constructive swarms based on predefined shapes [17]. This implements interactive guidance of the evolutionary development not unlike the immersive “gardening” techniques presented in [15]. The idea to train agents and deploy them during a game has been brought about on numerous occasions, e.g. in [13] or in Creatures (most recent release in 2003, Creature Labs).

$$f_{swarm} = \frac{1}{ia} \sum_i \sum_t \min(c(t, i), c_{max}), \text{ with } c_{max} = \frac{a}{t} \quad (1)$$

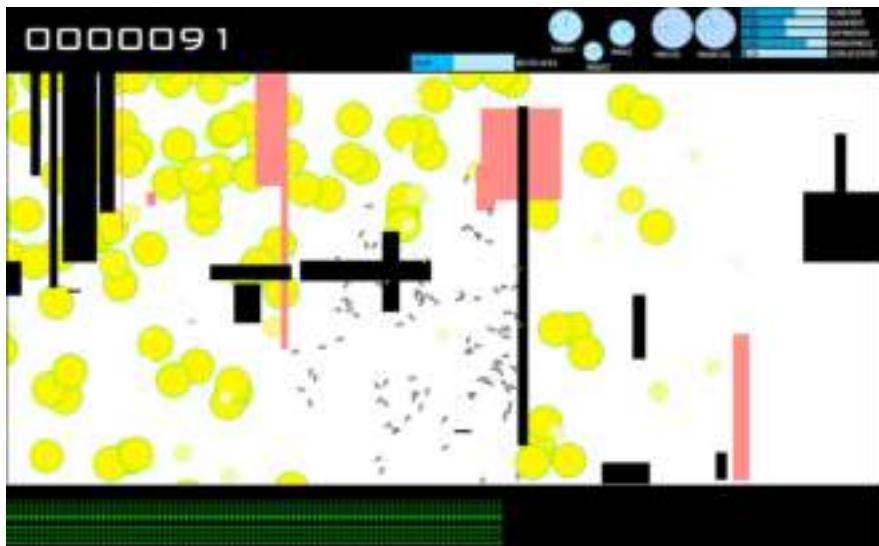


Fig. 5. Screenshot of the game “Crows in the Cornfield” in which the player maximizes the flock’s harvest by changing its configuration

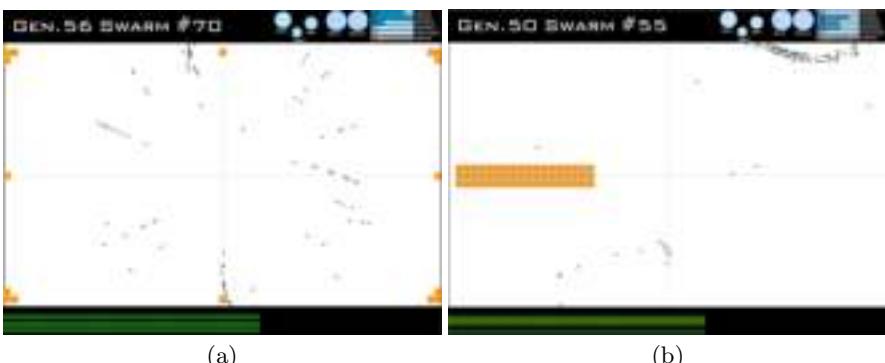


Fig. 6. (a) A flock has learned to swarm to the edges of the playground. (b) The flight in formation of a broad stripe maximizes the flock’s reward when hitting the tiles.

When entraining a swarm, the fitness function given in Equation 1 is applied to evaluate its performance. With a as the number of swarm agents, t the number of tiles, i the number of iterations and $c(t, i)$ as the number of collisions between swarm agents and a tile t at iteration i , the function can be read as follows: Over all iterations of a flocking simulation, each collision between an agent and a tile is considered for reward. However, in the case of a perfect distribution over the given tiles, there should only be up to c_{max} agents on one tile. The final sum is normalized by the product of iterations and the number of swarm agents.

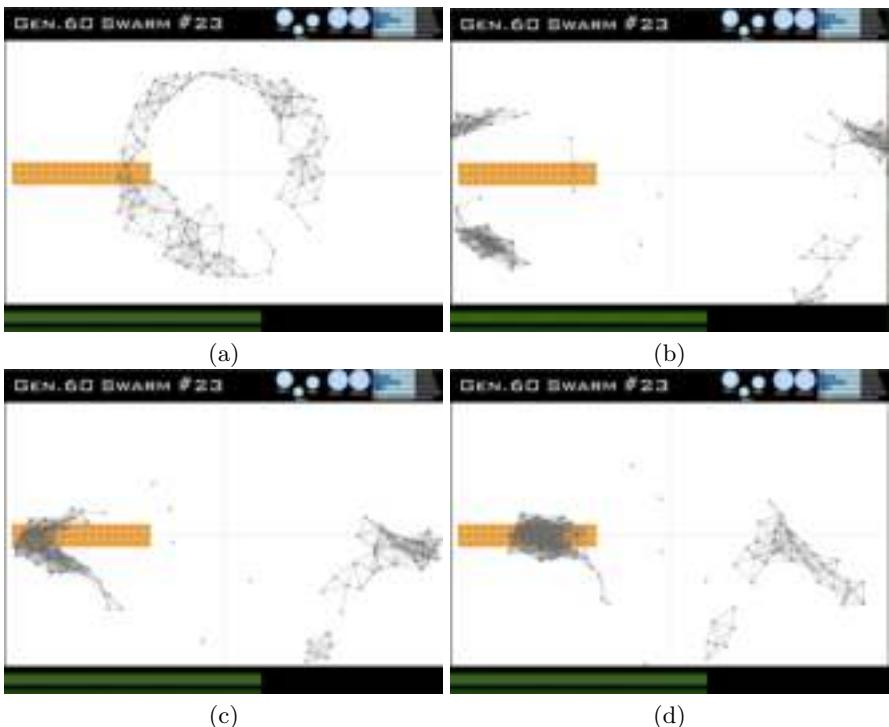


Fig. 7. An evolved swarm relies on interactions with the environment in order to hit a non-symmetrically placed tiled area

With the next generation composed of 50% recombined and 50% mutated material (with a mutation probability $p = 0.2$ on single flocking parameters), we successfully bred exemplary flocks. In Fig. 6(a) a swarm breaks up into several groups that are heading towards the edges of the playground, where they hit the tiles placed by the user. In another experiment, the flocking formation shown in Fig. 6(b) achieved a high fitness value. The similarity between the shape of the swarm formation in upper-right corner and the tiled area is notable. Another specimen of the same evolutionary experiment is presented in Figure 7. With the world center as the sole geometric point of reference, a solution to the given, non-symmetrical task is promoted that utilizes the general setting and a great degree of randomness. In Fig. 7(a) the flock radially expands from its origin. When repelled from the edges, it breaks into four parts (Fig. 7(b)). To the left and the right of the playground, two swarm groups re-emerge heading back towards the world center (Fig. 7(c)). This strategy results in an effective pass over the tiles to the left center of the playground (Fig. 7(c) and (d)).

5 Results

We have presented two games that rely on swarm-dynamics. A herding game is introduced in which the player effects the directed flocking of a swarm by

taking control of only one individual. This prototype tests the idea of player immersion in a dynamic, complex swarm system. With an obediently following swarm configuration the herding game poses a task as unambitious as finding the path through a maze from a top view. However, the challenge is increased by changing the swarm individuals' genotypes when flying over a mutation area. An intuitively operable GUI lends itself to the player to limit the damage. Control of the flock can be regained by patching single flocking parameters, by reconfiguring a group of swarm individuals to out-balance the maladjustment, or by canny flight maneuvers of the single individual that is manually steered by the player.

Second, a flock of virtual crows is controlled solely through manipulation of their interaction parameters in order to maximize their corn harvest. In addition to real-time adjustments of the swarm individuals as in the herding game, the player may breed assortments of swarm configurations upfront and deploy them afterwards in the game. This evolutionary component is especially important as designing specific flocking configurations can be a frustrating and tedious task.

6 Conclusion and Future Work

The fact that both presented games are embellished with additional game elements—for instance repelling blocks, deathtraps, time and score measures—cannot hide their prototypic character. Emphasis was put on testing ideas of player immersion in an artificial swarm. Both games live on an intuitive understanding of the swarms' flocking dynamics and on the players' skilled control. We believe that they have great potential to grow mature and to seemlessly incorporate many other gaming concepts, for instance realizing the swarm as a cooperative multi-player game.

As a next step, however, it would help to measure, rate and eventually improve the games in regards to basic aspects such as the means to control, (re-) configure and breed the swarms and the clarity of the immediate goals [14].

Swarms serve as a metaphor for highly dynamic complex systems [16]. As such, an improvement in understanding their intrinsic dynamics and in their control supports endeavors in all kinds of undertakings, whether they are economically or ecologically motivated. Therefore, we hope that by promoting the swarm metaphor in computer games, we foster the comprehension and mastery of other complex systems around us.

References

1. Aesthetics and Computation Group at the MIT Media Lab. Processing language and programming environment (October 2008), <http://processing.org/>
2. Blackwell, T.: Swarming and music. In: Miranda, E.R., Biles, J.A. (eds.) Evolutionary Computer Music, pp. 194–217. Springer, London (2007)
3. Blackwell, T.M., Bentley, P.: Improvised music with swarms. In: Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, pp. 1462–1467 (2002)
4. Burleigh, I.: Vigo: 3D: A framework for simulating and visualizing of three-dimensional scenes (October 2008), <http://vigo.sourceforge.net/docs/>

5. Freeman, W.T., Beardsley, P.A., Kage, H., Tanaka, K.I., Kyuma, K., Weissman, C.D.: Computer vision for computer interaction. *ACM SIGGRAPH Computer Graphics* 33(4), 65–68 (1999)
6. Jacob, C., Hushlak, G., Boyd, J., Nuytten, P., Sayles, M., Pilat, M.: Swarmart: Interactive art from swarm intelligence. *Leonardo* 40(3) (2007)
7. Jones, D.: AtomSwarm: A framework for swarm improvisation. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008*. LNCS, vol. 4974, pp. 423–432. Springer, Heidelberg (2008)
8. Klein, J.: breve: a 3D simulation environment for multi-agent simulations and artificial life (October 2008), <http://www.spiderland.org/>
9. Kwong, H., Jacob, C.: Evolutionary exploration of dynamic swarm behaviour. In: *Congress on Evolutionary Computation*, Canberra, Australia. IEEE Press, Los Alamitos (2003)
10. Nareyek, A.: Ai in computer games. *Queue* 1(10), 58–65 (2004)
11. Orkin, J.: Agent Architecture Considerations for Real-Time Planning in Games. In: *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment* (2005)
12. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. In: *SIGGRAPH 1987 Conference Proceedings*, vol. 4, pp. 25–34 (1987)
13. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation* 9, 653–668 (2005)
14. Sweetser, P., Wyeth, P.: Gameflow: a model for evaluating player enjoyment in games. *Comput. Entertain.* 3(3), 3–3 (2005)
15. von Mammen, S., Jacob, C.: Genetic swarm grammar programming: Ecological breeding like a gardener. In: Srinivasan, D., Wang, L. (eds.) *2007 IEEE Congress on Evolutionary Computation*, pp. 851–858. IEEE Press, Los Alamitos (2007)
16. von Mammen, S., Jacob, C.: The spatiality of swarms — quantitative analysis of dynamic interaction networks. In: *Proceedings of Artificial Life XI*, pp. 662–669. MIT Press, Cambridge (2008)
17. von Mammen, S., Jacob, C., Kókai, G.: Evolving swarms that build 3D structures. In: *IEEE Congress on Evolutionary Computation*, CEC 2005, Edinburgh, UK. IEEE Press, Los Alamitos (2005)
18. Wilensky, U.: The CCL at Northwestern University. Netlogo: A cross-platform multi-agent programmable modeling environment (October 2008), <http://ccl.northwestern.edu/netlogo/>

Fitness Diversity Parallel Evolution Algorithms in the Turtle Race Game

Matthieu Weber, Ville Tirronen, and Ferrante Neri

Department of Mathematical Information Technology,
University of Jyväskylä, P.O. Box 35 (Agora), FI-40014, Finland
`{matthieu.weber,ville.tirronen,neferran}@jyu.fi`

Abstract. This paper proposes an artificial player for the *Turtle Race* game, with the goal of creating an opponent that will provide some amount of challenge to a human player. *Turtle Race* is a game of imperfect information, where the players know which one of the game pieces is theirs, but do not know which ones belong to the other players and which ones are neutral. Moreover, movement of the pieces is determined by cards randomly drawn from a deck. The artificial player is based on a non-linear neural network whose training is performed by means of a novel parallel evolutionary algorithm with fitness diversity adaptation. The algorithm handles, in parallel, several populations which cooperate with each other by exchanging individuals when a population registers a diversity loss. Four popular evolutionary algorithms have been tested for the proposed parallel framework. Numerical results show that an evolution strategy can be very efficient for the problem under examination and that the proposed adaptation tends to improve upon the algorithmic performance without any addition in computational overhead. The resulting artificial player displayed a high performance against other artificial players and a challenging behavior for expert human players.

1 Introduction

In recent years, artificial game players have extensively been studied, see [1]. These games and thus the architecture of the artificial players can be classified into two groups: *complete information* games, when all players have the same information about the state of the game; *incomplete information* games, when each players has only an incomplete view of the game state.

Complete information games are also deterministic, thus a good player puts most of his efforts into foreseeing possible scenarios and movements the opponent may make in order to detect the most profitable move. Games of this kind have been widely studied by means of various artificial intelligence techniques. Some representative examples include from the simple tic-tac-toe [2] to the eminently complex go [3], [4], from classical chess and checkers [5] to the more recent sudoku [6].

In incomplete information games (see analogies and differences with the concept of imperfect information [7]), the player does not have a complete set of

information related to game state since, for example, he cannot know moves of the opponent, as in the classical case of the prisoner dilemma, or cannot predict moves the other players will make, as often happens in card games (this problem has been clear since the dawn of Artificial Intelligence, see [8]). Some examples of incomplete information games can be poker [9], bridge, backgammon, the ant wars [10]. In card games, each time a deck of cards is shuffled prior to the start of the game, a new game is settled. The overall process can, in this case, be seen as a system characterized by a state (i.e., what the players see on the table) plus some parameters (i.e., a deck of cards) which are affected by uncertainties.

This paper studies a complex game, namely *Turtle Race*, which is characterized by several uncertainties, including the unknown identity of the other players and proposes an artificial player for it. The core of this artificial player is a multi-perceptron neural network trained by means of a novel adaptive parallel evolutionary algorithm. This algorithm employs a multi-population and a refreshment of the individuals by means of migration controlled by a fitness diversity adaptation.

2 The Turtle Race Game

Schildkrötenrennen (*Turtle Race* in English) is a board game for two to five players by Reiner Knizia, initially published in 2004 by Winning Moves, Inc. Fig. 1 shows a picture of the game components. The game is composed of five wooden turtles in five different colors (red, green, blue, yellow and purple), five turtle tokens of the same colors, fifty-two cards and a board representing a linear track of ten squares. Each player is secretly assigned one color by randomly drawing one of the tokens, and all five turtles are placed on the first square of the board, called the *start square*. The goal of this game is to move one's turtle from the start square to the tenth square, called the *goal*. The cards are used for deciding how the turtles will be moved.

The players are dealt five cards each, and the remaining cards are placed face down, forming the draw deck. Players take turns until at least one turtle reaches the goal. On his turn, the player selects one card from his hand, moves one turtle



Fig. 1. Game components

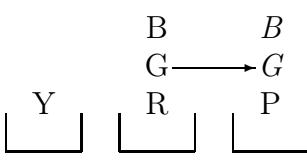


Fig. 2. When moving turtle G by one square forward, turtle B located on top of it moves at the same time

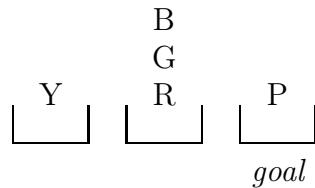


Fig. 3. To determine the winner, one examines P, R, G, and B in that order

based on the action symbolized on the card, and places the card face up on the discard pile. He then draws a new card from the draw deck. If the draw stack is empty, the discard stack is shuffled and replaces the draw deck.

Each card represents a turtle bearing a symbol. The color of the turtle determines which turtle will be moved, while the symbol defines in which direction and by how many squares it will move. Some cards represent a rainbow-colored turtle, meaning that the player must decide which turtle will be moved. The symbols are *plus*, *minus*, *double plus*, *arrow* and *double arrow*. *Plus*, *minus* and *double plus* respectively mean one square forward, one square backwards, and two square forward. The *arrow* and *double arrow* respectively mean one square forward and two squares forward, but can be applied only to the turtles which are the nearest to the start square. In the configuration depicted in Fig. 2 for example, *arrow* and *double arrow* cards can be applied only to turtle Y. The turtles bearing *arrow* or *double arrow* are always rainbow-colored. The cards are distributed as follows: 5 *plus*, 1 *double plus* and 2 *minus* for each color, 5 rainbow *plus*, 2 rainbow *minus*, 3 rainbow *arrow* and 2 rainbow *double arrow*.

Turtles can be stacked: as illustrated in Fig. 2, when moving one turtle, one also moves at the same time all turtles that are stacked on its back. When a turtle reaches a square already occupied by one or more stacked turtles, it is placed on top of that stack. The only exception is the start square, where the turtles are never stacked but placed side-by-side. The stacking mechanism allows a player, in some cases, to move his turtle even if he does not currently have a card of the right color in his hand, by moving another turtle situated below his own in the stack. Moreover, when playing an *arrow* or a *double arrow* card, if there is more than one turtle on the square nearest to the start, the player decides which one of these turtles he will move.

There are always five turtles in the game, even if there are less than five players. In this case, one or more turtles are *neutral*, i.e., not belonging to any player. When a turtle reaches the goal square, the game immediately ends and the players reveal their colors. It is possible that the turtle reaching the goal square is a neutral one. The rule for determining the winner is therefore as follows: the game winner is the player whose turtle is nearest to the goal square (possibly located exactly on this square) and is lowest in its stack. In other words, one considers the stacks of turtles from the goal rearward, and in each stack the turtles from bottom to top. The first non-neutral turtle is then the winner (see Fig. 3).

3 The Artificial Player

The core of the artificial player is a neural network, which associates a weight to each card held by the player. Rainbow-colored cards are decomposed into virtual single-colored cards, and a weight is associated to each one of them. The artificial player then sorts the cards according to their weights and the card with the lowest weight is selected to be played.

In order for the neural network to weigh a given card, the game logic considers the current game state, creates a copy of it and changes this copy according to the action depicted on the card. The neural network is then presented with these *before* and *after* game states and calculates the weight of that card based on this information.

The neural network chosen to be the core of the artificial player is a three-layer perceptron, using the hyperbolic tangent as the activation function [11]. This configuration has been chosen since it has shown a high performance with similar games, see [12] and [13]. The neural network is composed of nineteen neurons organized in three layers, has nineteen inputs and one output, and counts 325, real-number weights. Inputs of the neural network are the artificial player's own color, a synthetic representation of the game state before playing a given card and the representation of the game state after playing that card. Both representations have the same parameters:

- The position on the board of the artificial player's turtle, as an offset from the start square (integer number between zero and nine) and its position in the stack, as an offset from the bottom of the stack (integer number between zero and four). These two parameters give the neural network a global view of its position in the game.
- The number of turtles situated above and below the artificial player's within the stack (as integer numbers between zero and four).
- The number of turtles situated on the square preceding the artificial player's, i.e., nearer to the start square, the number of turtles situated on the square following that of the artificial player's, and the number of turtles situated on the second square after that of the artificial player's (integer numbers between zero and four).
- The number of turtles situated on the start square (integer number between zero and five).
- The number of squares between the artificial player's turtle and the goal square (integer number between one and nine).

The artificial player design consists of detecting a set of 325 weights of the neural network, ensuring a high performance in playing the turtle game.

It can be seen that except for its own color, the neural network is fed with no further information about the color of the other turtles. The reason is that a player ignores the color of the various opponents as well as the color of the neutral turtles. This fact leads to the consequence that a player (the neural network in this case) makes a decision on the basis of those cards he can see (his own cards) and possible configurations which are visible on the board. Thus, it is

clear that, due to the nature of the game, given a configuration of turtles on the track, success of the decision made by the artificial player is heavily influenced by parameters that cannot be controlled, i.e., cards held by the other players and the way the deck has been shuffled. consequently, evaluation of a playing strategy in a single match is not reliable since “good luck” and “bad luck” play an important role in this game. This phenomenon can be seen as noise unavoidably introduced into the system. Thus, in order to perform proper ranking of two players a certain amount of games are required and the quality of a player can be extracted by means of a statistical analysis on the success of the players under consideration.

Thus, the quality of playing strategy has been performed in this paper by means of the following method. Let a and b be two players and N the number of games played between a and b . If a is the playing strategy proposed by the aforementioned neural network, the objective function to be minimized is then defined as:

$$F(a, b) = 1 - \frac{V_a}{V_a + D_a} \quad (1)$$

where V_a is the number of victories of a against b , and D_a the number of defeats of a against b . The value of N has been fixed to 100, which has been found to be a good trade-off between statistical significance and speed of execution. It must be noted that $V_a + D_a$ is not necessarily equal to N , since there are cases where a neutral turtle reaches the goal square while both players’ turtles are still on the start square. There is no rule in the game to determine the winner in such a situation.

The opponent player for the neural network is a random player, i.e., a player which randomly performs its choices during the game time. This choice is based on the assumption that the neural network will challenge a large amount of random players and eventually develop a strategy that allows a robust behavior against other players which have an actual playing strategy (e.g., human players). This assumption is limited to such games as Turtle Race which have an inherently large amount of randomness in their structure. This choice can be further justified by the theoretical results given in [14]. It is shown that a complex and expert player can be replaced by a simple random player which executes multiple games. This fact based on the property of the random player’s average behavior, given a large enough number of trials, tends to asymptotically converge toward the behavior of an expert player. This property has already been used when developing so-called *Monte Carlo* players for various games, see [15]. Although not identical, the use of a random player as an opponent when training a neural network-based player is strongly correlated to its use in Monte Carlo players, which has been confirmed experimentally.

The problem studied in this paper is then the minimization of the fitness function F in eq. (1) dependent on the 325 weights which characterize neural network a . These weights vary in a decision space $D =]-\infty, \infty[^{325}$. As shown above, the quality of a candidate neural network is identified by its success rate against 100 random players.

4 Fitness Diversity Parallel Evolutionary Algorithms

In order to perform the neural network training, four evolutionary algorithms have been tested. More specifically the following algorithms have been considered: 1) the evolutionary algorithm proposed in [5] for a neural network training, indicated here with Checkers Algorithm (CA), 2) a standard Differential Evolution (DE) [16], [17] with the so called DE/rand/1 strategy, 3) a Particle Swarm Optimization (PSO) according to the implementation proposed in [18] for a similar neural network training, and 4) an Evolutionary Strategy, see [19] and [20], employing the so called “plus strategy” (ES+), i.e., the offspring solutions are merged with the parent solution and the survivor selection scheme selects a predetermined amount of candidate solutions having the best performance and uncorrelated mutation with n step sizes, i.e., a step size is sampled for each design variable.

In order to evaluate the performance of each candidate solution, 100 games against 100 random players are performed and the fitness value is computed as shown in eq. (1). For those evolutionary algorithms which allow an individual to survive over several generations (e.g., DE or ES+), the fitness value is updated, taking into account run off of the previous matches.

At the end of each generation, the range of variability of the fitness values is calculated for each population:

$$\gamma = F_{worst} - F_{best} \quad (2)$$

where F_{worst} and F_{best} are the worst and best values of the population, respectively. The index γ can be seen as a fitness diversity index, see e.g. [21], [22], [23], and [24], which is designed for this specific co-domain (the fitness values vary between 0 and 1). The index γ varies between 0 and 1; if $\gamma \approx 1$ the population is likely to be highly diverse since performance of the population is spread out over the whole range of variability of the co-domain; on the contrary, if $\gamma \approx 0$ performance for the candidate neural networks in the population is very similar and the diversity is therefore low. In order to enhance exploration of the algorithmic system and prevent undesired conditions of stagnation and premature convergence, a migration mechanism has been introduced by employing information of the fitness diversity. More specifically, if in a population $\gamma < 0.05$, 10 individuals are pseudo-randomly swapped between this population and the population with the lowest diversity value. This system allows a harmonic development of the parallel algorithm and, most importantly, performs a refreshment of genotypes in the various populations. The groups of swapped individuals are likely to have similar performance, but by means of a different strategy (genotypically different) since they evolved within different populations. The presence of new genotypes in the population will then be beneficial, since it will allow a recombination between individuals which are strong and (probably) genotypically distant, see for analogy [25] and [26].

5 Numerical Results

The CA is based on [5], with $\sigma = 0.05$. However, some changes have been made in the parent selection scheme in order to adjust the CA to the problem at hand. A linear ranking [27] with a selective pressure $sp = 1.7$ has been included. The DE, as described in [16], has been implemented with $C_r = 0.3$ and $F = 0.7$. The Particle Swarm Optimization (PSO), as described in [18], has been implemented with parameters $\phi_1 = 2$ and $\phi_2 = 2$. ES+ was implemented, as described in [20], with $\lambda = \mu$ and initial values of mutation step size $\sigma_i = 0.1$. Each algorithm has been run with a population size of 30 individuals; each was run for 35000 fitness evaluations. Each algorithm has been run twice with 10 parallel populations each (i.e., a statistic set of 20 evolving populations). The same algorithms have also been run without the adaptive migration mechanism: each algorithm has been run for 20 independent runs. In order to perform a fair comparison, the same initial populations used for the fitness diversity parallel evolutionary algorithms have also been employed for this experiment on isolated populations.

Fig. 4 shows the performance trend (averaged over the 20 populations) of the proposed fitness diversity parallel evolutionary algorithms. In order to enhance clarity in the result representation, the results have been averaged within intervals of 600 fitness evaluations. Fig. 5 shows the average performance trend (over 20 independent runs) of the four algorithms under study in the isolated implementation. Table 1 lists the average final fitness values with the corresponding standard deviation values.

As displayed in Table 1, ES+ obtained the best performance solution algorithm, while CA succeeded in performing only a marginal improvement. DE seems to be the second best and PSO comes third. The migration process seems to be beneficial for all algorithms in a similar way. As a matter of fact, it can be observed that ranking among the algorithms remains the same whether migrations are allowed or not.

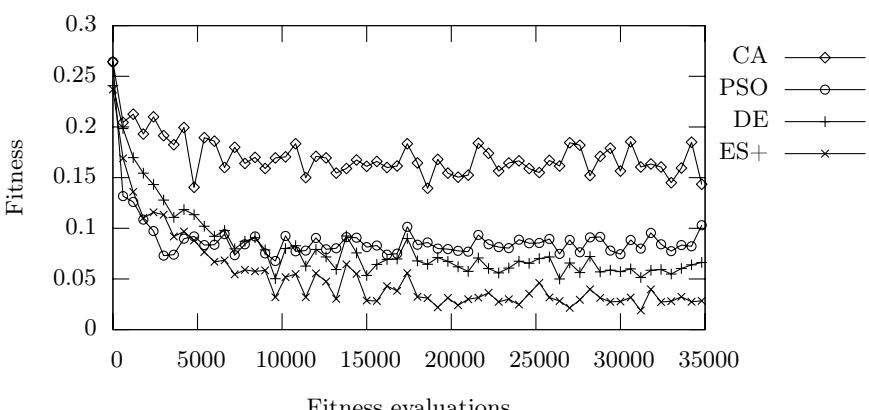


Fig. 4. Performance trend for the four algorithms with migration

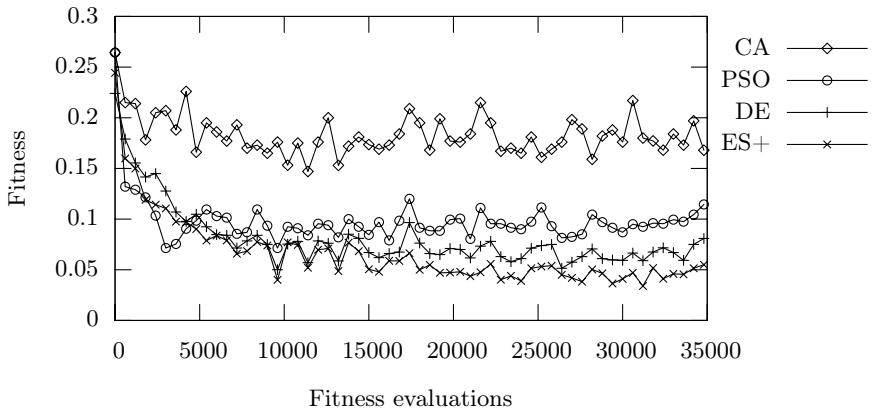


Fig. 5. Performance trend for the four algorithms without migration

Table 2 shows the performance of the best players trained according to the four algorithms with and without migrations. Each of the eight players competes in 10,000 games against all of the other seven players as well as against itself, and the score of the first player is given as a percentage of victories. One can see that the results of this cross-comparison is consistent with the results observed in Table 1, when the neural network-based players were confronted with random players.

Finally, the best player trained with ES+ and migration has been playing ten games against five expert, human players. The results are shown in Table 3. The artificial player won on average 4.4 games, with a standard deviation of 1.02, proving to be a fairly challenging player.

Table 1. Average fitness and standard deviation for all four algorithms, with and without migrations

Algorithm	Fitness
Migration	
CA	0.1435 ± 0.0414
PSO	0.1030 ± 0.0282
DE	0.0663 ± 0.0123
ES+	0.0280 ± 0.0066
Isolated	
CA	0.1680 ± 0.0407
PSO	0.1145 ± 0.0213
DE	0.0809 ± 0.0163
ES+	0.0550 ± 0.0084

Table 2. Cross-comparison of the best individuals produced by each algorithm. The values are expressed in percentage of victories of the first player over 10,000 games.

		Second Player								
		Migration				Isolated				
First Player	Migration	CA	PSO	DE	ES+	CA	PSO	DE	ES+	
	Isolated	50.2	45.8	42.7	32.1	53.3	53.3	44.1	33.7	
Player	Migration	CA	51.4	45.1	46.7	33.4	55.6	55.7	46.2	35.6
	Isolated	PSO	59.3	54.8	55.5	41.3	64.2	62.6	53.8	42.5
First Player	Migration	CA	67.8	64.3	66.1	51.8	72.1	75.0	66.1	51.0
	Isolated	PSO	46.7	41.1	37.7	28.7	49.9	54.4	40.7	31.7
First Player	Migration	DE	45.1	37.3	40.4	25.0	44.8	47.3	37.5	28.1
	Isolated	ES+	57.4	50.5	51.7	37.4	60.1	61.0	52.0	38.8
First Player	Migration	ES+	66.4	61.4	62.0	49.1	68.8	70.7	63.8	49.8

Table 3. Number of victories of the artificial player trained with ES+ and migrations against five human players over ten games

Opponent	Player 1	Player 2	Player 3	Player 4	Player 5
Victories	3	4	6	5	4

6 Conclusion

An artificial, neural network-based player for the *Turtle Race* game has been presented. The neural network has been trained by means of a novel parallel evolutionary algorithm which employs a fitness diversity measure to activate a migration of the individuals between populations in order to perform a refreshment of the genotypes and prevent premature convergence and stagnation. Four popular evolutionary algorithms have been tested within the proposed adaptive parallel framework. Numerical results show that an Evolution Strategy employing a “plus” strategy (despite the presence of noise) in the survivor selection scheme is fairly promising for this class of problems. In addition, the proposed fitness diversity migration seems to be beneficial to the tested algorithms regardless of the evolutionary structure which characterizes them. A test carried out against expert human players shows that the designed artificial player is rather challenging since it succeeds in defeating human players in almost half of the games.

References

1. Lucas, S., Kendall, G.: Evolutionary computation and games. *IEEE Computational Intelligence Magazine* 1, 10–18 (2006)
2. Fogel, D.: Using evolutionary programming to create networks that are capable of playing tic-tac-toe. In: *Proceedings of IEEE International Conference on Neural Networks*, pp. 875–880 (1993)
3. Richards, N., Moriarty, D., Miikkulainen, R.: Evolving neural networks to play go. *Applied Intelligence* 8, 85–96 (1998)
4. Runarsson, T.P., Lucas, S.M.: Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board go. *IEEE Transactions on Evolutionary Computation* 9, 628–640 (2005)
5. Chellapilla, K., Fogel, D.: Evolving an expert checkers playing program without using human expertise. *IEEE Transactions on Evolutionary Computation* 5, 422–428 (2001)
6. Moraglio, A., Togelius, J.: Geometric particle swarm optimization for the sudoku puzzle. In: *GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 118–125 (2007)
7. Fudenberg, D., Tirole, J.: *Game Theory*. MIT Press, Cambridge (1993)
8. Barricelli, N.A.: Esempi numerici di processi di evoluzione. *Methodos*, 45–68 (1954)
9. Barone, L., While, L.: Evolving adaptive play for simplified poker. In: *Proceedings of IEEE Intl. Conf. on Computational Intelligence (ICEC 1998)*, pp. 108–113 (1998)
10. Jaśkowski, W., Krawiec, K., Wieloch, B.: Evolving strategy for a probabilistic game of imperfect information using genetic programming. *Journal Genetic Programming and Evolvable Machines* 9, 281–294 (2008)

11. Engelbrecht, A.P.: Computational Intelligence: An Introduction. John Wiley & Sons Ltd., Chichester (2002)
12. Bourg, D.M., Seemann, G.: AI for Game Developers. O'Reilly, Sebastopol (2004)
13. Chellapilla, K., Fogel, D.: Evolution, neural networks, games, and intelligence. *Proceedings of the IEEE* 87, 1471–1496 (1999)
14. Abramson, B.: Expected-outcome: a general model of static evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 182–193 (1990)
15. Bürgmann, B.: Monte carlo go. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 182–193 (1993)
16. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal on Global Optimization* 11, 341–359 (1997)
17. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution—A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
18. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948 (1995)
19. Rechemberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution. Fromman-Hozlboog Verlag (1973)
20. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation, pp. 71–87. Springer, Berlin (2003)
21. Caponio, A., Casella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives. *IEEE Transactions on System Man and Cybernetics-part B* 37, 28–41 (2007)
22. Neri, F., Toivanen, J., Casella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing HIV multidrug therapies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4, 264–278 (2007)
23. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: An enhanced memetic differential evolution in filter design for defect detection in paper production. *Evolutionary Computation* 16(4), 529–555 (2008)
24. Caponio, A., Neri, F., Tirronen, V.: Super-fit control adaptation in memetic differential evolution frameworks. *Soft Computing—A Fusion of Foundations, Methodologies and Applications* (to appear, 2009)
25. Smith, J.E.: Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 37, 6–17 (2007)
26. Zamuda, A., Brest, J., Bošković, B., Žumer, V.: Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In: *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 3719–3726 (2008)
27. Back, T.: Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: *Proceedings of the IEEE World Congress on Computational Intelligence*, vol. 1, pp. 57–62 (1994)

Evolving Strategies for Non-player Characters in Unsteady Environments

Karsten Weicker¹ and Nicole Weicker²

¹ HTWK Leipzig, Germany

weicker@imn.htwk-leipzig.de

² College of Education PH Heidelberg, Germany

weicker@ph-heidelberg.de

Abstract. Modern computer games place different and more diverse demands on the behavior of non-player characters in comparison to computers playing classical board games like chess. Especially the necessity for a long-term strategy conflicts often with game situations that are unsteady, i.e. many non-deterministic factors might change the possible actions. As a consequence, a computer player is needed who might take into account the danger or the chance of his actions. This work examines whether it is possible to train such a player by evolutionary algorithms. For the sake of controllable game situations, the board game Kalah is turned into an unsteady version and used to examine the problem.

1 Motivation

Classical games like chess, checkers, or nine men's morris offer clearly defined game configurations and a fixed set of rules. For this reason, the minimax algorithm is a proper “intelligence” for an opponent played by the computer—provided that the situations on the board can be properly assessed in an ongoing game.

Modern computer games like strategy video games are characterized by a high degree of interaction with the human player(s). The number of states exceeds that of classical games by far—analogously the branching degree of the decision tree. Due to additional time restrictions, the computation of the next action is a real-time application.

Although we deal with a variation of a classical board game in this paper, we want to motivate this work by looking at the challenges developers of modern computer games have to face when implementing strategies for non-player characters.

Variety concerning the actions taken: Games should generate a variety of game situations non-deterministically—caused by random events, the actions of the non-player character, and the broad spectrum of actions of the human player. This increases the complexity considerably that must be accounted for by the programmers. The game “Lufia II” is a negative example where always the same events occur (except one random incident).

Real-time behavior: Turn-based games are always subject to certain restrictions concerning the design of the possible actions—especially long-term actions must be divided into several partial steps. Programmers benefit from the synchronization points imposed by turns. However with increasing computer performance real-time behavior is getting more interesting even in complex games since the reality can be imitated better and the human player can act more flexibly. An example for turn-based games are the fights in “Heroes of Might and Magic”. “Homeworld” is an example for a real-time game.

Flexible Intelligence: Human players expect non-player characters to act reasonably and intelligently. However, since each players freedom of action is restricted by his opponents’ actions, the intelligence must take the situation of the opponent into consideration. Especially long-term actions should be adapted if the circumstances change. Such an ample scope within a game is called *unsteady environment* within this work. A negative example concerning flexible intelligence is the game “StarCraft” in which a non-player character may be blocked by closing a passage way—the computer strategy is not able to change its plan. A related problem is the anticipation of the computer that the human player will act in a specific way. For example in “Heroes of Might and Magic” it is possible to defuse a critical situation by absurd actions. Altogether many computer games are so complex that the developers cannot consider all situations in their programming. This leads to interesting effects like in “Jedi Knights III” where it is possible to “jump out of the game”.

Unsteady environments are the subject of this examination. The scenario in figure 1 demonstrates the importance for highly interactive real-time games: The long-term goal is to plan a path for a knight to a castle. However the environment is unsteady since the opponent might disturb the shortest path. As a consequence the knight’s decision to visit the castle should take the alternative longer path into consideration, too.

Definition 1. *An unsteady environment is given if the presumed future game situations (e.g. in a minimax search for turn-based games) might not fit the real game situations. The possible difference Δ between the situations is referred to as degree of unsteadiness.*

In the bottom line, heuristic strategic planning is necessary. There are three general approaches for impending unsteady situations.

- using a procedure which works well for safe situations—in the example above this could be the direct path
- a “more weak” procedure for safe situations—e.g. by planning a slightly disordered path
- a procedure that is trained specifically for unsteady situations—e.g. by taking the distance to the opponents’ troops into consideration.

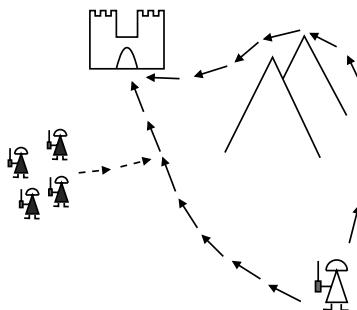


Fig. 1. The direct path to the castle is unsteady because of possible actions of the opponent. If this was the only gateway, a different long-term goal could be more sensible. However, since there is an alternative less risky path, visiting the castle is an acceptable goal.

2 Kalah as Object of Investigation

To examine the situation stated above a clearly defined test environment is necessary. We have to consider the following factors:

- there are easily detectable strategies,
- the problem is more challenging than most quickly developed scenarios, and
- the problem is not as difficult as chess or Go.

As a consequence we selected the simple board game Kalah (Mancala) since it fits well concerning the degree of difficulty. Additionally, it is possible to adjust the degree of unsteadiness continuously.

Kalah denotes a broad class of African board games, in which tokens are redistributed on the board according to certain rules. The goal is to gather the majority of the tokens in one's own safe place. In the version in figure 2 the big pit to the right is the Kalah, the safe place for the player in front of the board.

The rules of the game are summarized quickly.

- Alternately the players clear a pit in front of them and redistribute the tokens in counter-clockwise direction—including the own Kalah but ignoring the opponent's Kalah (see figure 3).
- If the last token hits an empty pit in front of the player, the token and all tokens of the corresponding opposite pit are transferred into the own Kalah (figure 4).
- If the last token hits the own Kalah, the player must clear another non-empty pit and distribute the tokens (figure 5).
- The game ends when the player who has his turn has no tokens left in his pits. Often all tokens in the opponent's pits are assigned to the opponent's Kalah. However, we chose the alternative and transfer the opponent's tokens into the finishing player's Kalah (figure 6). The degree of difficulty appears to be similar in both versions but different strategies are necessary towards the end of the game.

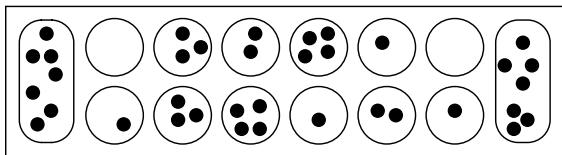


Fig. 2. Kalah board with 36 tokens and a fictitious game situation. Each player can act on the pits in front of him. And the pit to the right is the Kalah—the tokens in there correspond to the own score.

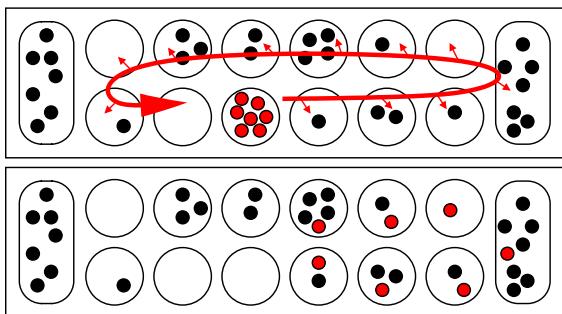


Fig. 3. Basic move: The tokens of an own pit are redistributed in counter-clockwise direction including the own Kalah and ignoring the opponent's Kalah

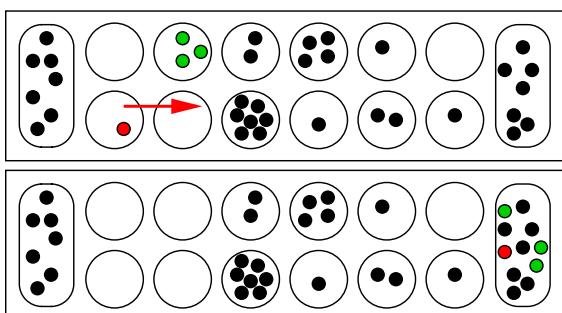


Fig. 4. Plundering the opponent: If the last token hits an own empty pit, the token and the tokens in the corresponding opponent's pit are transferred to the own Kalah

Kalah(m, n) denotes the configuration of the game with m pits for each player and n initial tokens in each pit. We consider the configuration Kalah(6,6). In his diploma thesis Fesser (2004) has investigated the complexity of the game by an empirical estimation of the size of the game decision tree. He states that Kalah(6,6) has a typical tree size in the range of $10^{26} - 10^{31}$ (with Alpha-Beta-pruning). Kalah(5,5) has a tree size in a range of $10^{18} - 10^{22}$. Altogether Kalah is

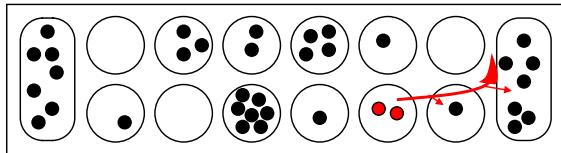


Fig. 5. Keep going: If the last token hits the own Kalah, another pit must be cleared and the tokens redistributed

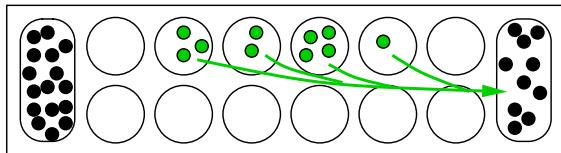


Fig. 6. End of game: If all pits of the player who has his turn are empty, the game is over. In our version the player with the empty pits gets all tokens left in the opponent's pits.

a rather simple game: Irving et al. (2000) state that an optimal gaming strategy is known for those variants with either $m \leq 6$ and $n < 6$ or with $m < 6$ and $n \leq 6$.

Because simple Kalah is considered as solved, other investigations switch to more complex Mancala variants like Bao (Donkers et al., 2004). However, in our context of unsteady environments this simple version is the first choice because the unpredictable situations are the challenging factor for the game strategies.

We introduce unsteadiness by the following mechanism. After each changeover, tokens are transferred from one pit to another pit with probability $p \in [0, 1)$. The maximum number of transferred tokens is defined by $c \in \mathbb{N}$. As a consequence, (p, c) controls the degree of unsteadiness.

3 Computer Players for Kalah

In order to generate “intelligent” behavior of the computer player we use the established minimax search in a decision tree—for the purpose of this work we use the relatively small search depth of four. This means that we consider all possible sequences of four moves beginning with a computer player’s move and ending with an opponent’s move after three changeovers. Note that a player’s move could consist of cleaning out several pits according to the rules above. Within the context of this feasibility study the small search depth enables extensive optimizations within reasonable time. Alpha-Beta pruning could be used to omit certain “bad” move sequences, however we decided to abandon this option. All game situations at depth 4 are rated and the first decision is based on the assumption that we maximize our benefit and the opponent tries to mini-

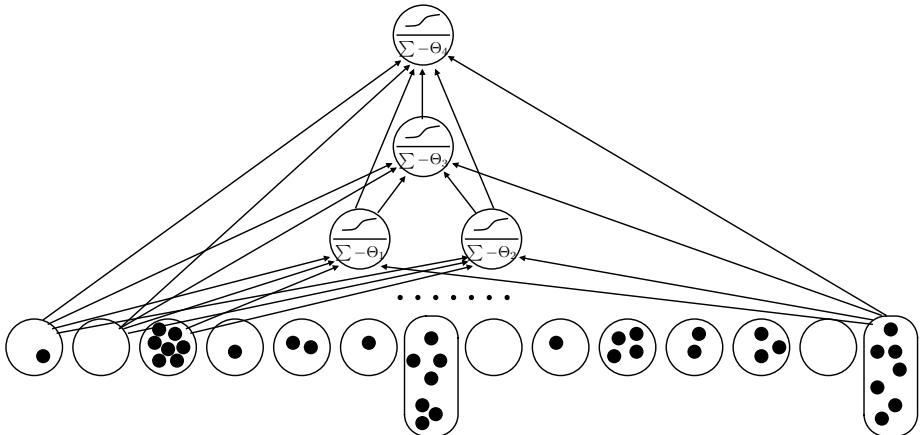


Fig. 7. Neural network to rate a game situation

mize it. All strategies described in the remainder differ in the rating of the game situations only.

For safe Kalah, the “piece difference information” player (PDI) is an approved strategy that considers the difference of the tokens in the Kalahs solely. More precisely, the following rating is used within our PDI player:

- a lost game is rated with $-1 + \frac{\text{ownTokens}}{100}$,
- a won game is rated with $1 + \frac{\text{ownTokens}}{100}$ and
- a situation in an ongoing game is rated with

$$\frac{\text{ownTokens}}{\text{ownTokens} + \text{opponentTokens}}. \quad (1)$$

This is a fairly good strategy although strategic planning is only possible where an immediate gain of tokens is reached within the search depth of minimax.

A more “weak” game strategy is generated using a function that computes a rating from the numbers of tokens in all pits. This function replaces the formula in equation (1) and enables more strategic considerations going beyond mere profit maximization. In our approach, the function is represented by a neural network with two hidden layers with two respective one neuron, short-cut connections and a sigmoid activation function. An exemplary net is shown in figure 7. The resulting value of the net is always an element of $[0, 1]$.

In the remainder of the article we will adapt the neural network by evolution strategies for playing in two different scenarios: safe and unsteady Kalah.

4 Evolutionary Search for Strategies

Evolutionary algorithms are often used to calibrate the rating function of game strategies. A well-known example is the checkers player by Chellapilla and Fogel (2000). Also backgammon (Pollack and Blair, 1998) and Go (Lubberts and Miikkulainen, 2001) were tackled successfully. For Kalah a broad range of techniques has been used: neural networks (Davis and Kendall, 2002; Fesser, 2004), genetic programming (Ahlschwede, 2000) and sets of rules (Fesser, 2004).

In our work neural networks are optimized by an evolutionary algorithm using the evolution strategy mutation (Schwefel, 1995) with one strategy parameter σ . The population size of 10 neural networks is comparatively small. The evolutionary loop is borrowed from evolutionary programming: Each individual is mutated exactly once and the binary n -stage tournament selection is used on parents and offspring. However, the tournament selection is modified as follows: Each individual plays six matches as opening player and six matches as reacting player against randomly chosen opponents from the population. In addition, two matches are played against two individuals that are stored in a “hall of fame” (Rosin, 1997; Lubberts and Miikkulainen, 2001) (for details see below). Each individual receives the following points for a match:

$$\begin{aligned} \text{Win: } & 2 + \frac{\text{ownTokens}}{100} \\ \text{Tie: } & 1.5 \\ \text{Reverse: } & \frac{\text{ownTokens}}{100} \end{aligned}$$

These values favor a strategy with many ties slightly over a strategy where a win is matched by a reverse. We chose this set-up since in Kalah two approximately equal players should be able to reach a tie.

The “hall of fame” contains 10 individuals with good performance from previous generations. In each generation the best individual enters the hall and replaces the individual that has already played at least in five matches as a “hall of fame” member and that shows the worst percentage of won games.

As in many reports in the literature, the tournament selection proves of value in our work. The single-population coevolution is able to generate a wide set of different strategies and, in the long run, very flexible computer players may result. In addition, it is valuable that the PDI player is not used within the training of our strategies. As a consequence the PDI player is used as reference player to assess the developed strategies.

The set-up of the hall of fame and the parameters of the evolution strategy have not been tuned since the focus of this paper is the feasibility of adapting a strategy to an unsteady environment.

5 Experiments

In a first preliminary study, we investigated whether the approach is able to produce a competitive strategy at all. The safe version of Kalah was used in

Table 1. Analysis of the “hall of fame” players after the last generation: The number of won games out of 10 games as opening and reacting player

	individual									
	1	2	3	4	5	6	7	8	9	10
as a opening player	9	9	8	6	4	3	2	1	1	0
as reacting player	7	3	3	0	2	1	1	0	0	0

the evolution of strategies for 200 generations. Afterwards the 10 players in the “hall of fame” compete against the PDI player in 20 matches each. The results are shown in table 1. They indicate that the mechanism with the “hall of fame” is a practicable technique to gather at least a few good players. In this experiment we used neural networks without threshold values in the neurons. In the successive investigations we also evolved a threshold value in each neuron—all other parameters except the number of generations (800) have been unchanged.

The unsteady version of the game was generated using $(p, c) = (0.4, 3)$. The number of the tokens moved from one pit to another pit is determined by a random number: with probability 0.7 one token is moved, two tokens with probability 0.2 and three tokens with probability 0.1. As a consequence the degree of unsteadiness is rather high.

The best evolved strategies, called SNN for the safely evolved player and uNN for the player trained in an unsteady environment, are tested against the PDI player as well as in direct confrontations. The results are shown in table 2. It can be clearly seen, that the best unsteady evolved player performs better than the safely evolved one. Merely in a direct confrontation the safe player wins one game more but gains less tokens totally. Note, the direct confrontation of two players using neural networks leads to a deterministic behavior in the safe environment. This is due to the fact that neural networks assign almost always unique values to the different situations and, as a consequence, the same course occurs in all 40 iteratively played matches. This is not the case with the PDI player where different moves/situations are assessed with identical values which leads to a random choice between those alternatives.

In addition, a statistical hypothesis test is carried out to find out whether there is a statistically significant difference between two players tackling the same opponent—here the number of tokens in the Kalah is used as a criterion within the test. The results are displayed in table 3.

The resulting picture is mixed. On the one hand uNN dominates sNN when playing against the PDI. On the other hand, the direct competition is rather balanced in unsteady environments.

Nevertheless, we can conclude the following statements.

1. It is possible to use an unsteady environment for a successful training of a neural network based player.
2. In spite of sparse computation time the player evolved in an unsteady environment is able to beat the PDI player in approx. 70% of the matches within the unsteady environment.

Table 2. Results for 40 games as opening player and 40 games as reacting player. Each entry shows the number of won games and the average number of tokens gathered in the Kalah. “sNN” denotes the safely evolved player and “uNN” the unsteadily evolved player.

		safe environment			unsteady environment		
		opening	reacting	total	opening	reacting	total
uNN : PID	# won	28	28	56	31	25	56
	E[tokens]	38.03	37.85	37.94	38.93	36.95	37.94
	V[tokens]	17.07	12.68	14.88	16.77	21.60	20.16
uNN : sNN	won	40	40	80	21	18	39
	E[tokens]	deterministic			36.75	35.63	36.19
	V[tokens]				26.73	16.38	21.88
sNN : PID	won	28	19	47	22	14	36
	E[tokens]	36.35	34.88	35.62	35.68	34.20	34.94
	V[tokens]	18.58	22.31	20.98	35.17	26.56	31.41

Table 3. Hypothesis tests concerning the gathered tokens in the games. For the compared strategies the tokens against the third strategy are considered. In each field the significant strategy has been entered. Otherwise “n.sig.” indicates missing significance.

		opening	reacting	total
safe:	uNN : sNN	uNN	uNN	uNN
unsteady:	uNN : sNN	uNN	uNN	uNN
	uNN : PDI	n.sig.	n.sig.	n.sig.
	sNN : PDI	sNN	n.sig.	sNN

3. Even more the player evolved in the unsteady environment shows an equally good performance against the PDI player in the safe environment.
4. For broader conclusions more computation time for the training and a statistical evaluation based on several evolved players is necessary.

6 Conclusion and Outlook

The development of flexible “intelligent” computer players is an important research field for computer game developers. In this paper it is shown that it is possible to use neural networks trained by evolutionary algorithms to tackle unsteady environments—at least for the simple scenario of the game Kalah.

The presented results are certainly only a first “scratch” on the surface of strategies for unsteady environments. Nevertheless Kalah has turned out to be an interesting object for such investigations.

A significantly bigger number of experimental data is necessary to prove empirically whether there is an qualitative difference between safely and unsteadily evolved players. First steps towards an objective standard of evaluation are mandatory. Also, future examinations have to deal with the influence of the degree of unsteadiness as well as the parameter settings for the neural networks and the evolutionary algorithms.

A long-term goal is to switch from neural networks to an approach using sets of rules for the assessment of game situations. This would enable a concrete analysis which rules are of more importance when increasing the degree of unsteadiness. And as a consequence we could imagine an adaptive player that changes the set of rules dynamically when he senses a higher risk of unsteadiness.

Bibliography

- Ahlschwede, J.: Using genetic programming to play mancala (2000),
<http://www.corngolem.com/john/gp/index.html>
- Chellapilla, K., Fogel, D.B.: Anaconda defeats hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In: Proc. of the 2000 Congress on Evolutionary Computation, Piscataway, NJ, pp. 857–863. IEEE Press, Los Alamitos (2000)
- Davis, J.E., Kendall, G.: An investigation, using co-evolution, to evolve an awari player. In: Fogel, D.B., El-Sharkawi, M.A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M. (eds.) Proc. of the 2002 Congress on Evolutionary Computation, Piscataway, NJ, pp. 1408–1413. IEEE Press, Los Alamitos (2002)
- Donkers, J., van den Herik, H.J., Uiterwijk, J.W.H.M.: Probabilistic opponent-model search in bao. In: Rauterberg, M. (ed.) ICEC 2004. LNCS, vol. 3166, pp. 409–419. Springer, Heidelberg (2004)
- Fesser, M.: Entwicklung von Spielstrategien mittels evolutionärer Algorithmen am Beispiel von Kalaha. Master's thesis, Universität Stuttgart, Formale Methoden der Informatik. Diplomarbeit Nummer 2184 (2004)
- Irving, G., Donkers, J., Uiterwijk, J.: Solving Kalah. ICCA Journal 23(3), 139–147 (2000)
- Lubberts, A., Miikkulainen, R.: Co-evolving a Go-playing neural network. In: Coevolution: Turning Adaptive Algorithms upon Themselves. Birds-of-a-Feather Workshop, Gecco 2001, San Francisco, CA, pp. 14–19 (2001)
- Pollack, J.B., Blair, A.D.: Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. Machine Learning 32(3), 225–240 (1998)
- Rosin, C.D.: Coevolutionary Search Among Adversaries. PhD thesis, University of California, San Diego (1997)
- Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley & Sons, New York (1995)

Grid Coevolution for Adaptive Simulations: Application to the Building of Opening Books in the Game of Go

Pierre Audouard¹, Guillaume Chaslot², Jean-Baptiste Hoock³, Julien Perez³,
Arpad Rimmel³, and Olivier Teytaud³

¹ Go Expert

² Maastricht University

³ TAO (Inria), LRI, UMR 8623(CNRS – Univ. Paris-Sud),
bat 490 Univ. Paris-Sud 91405 Orsay, France

Abstract. This paper presents a successful application of parallel (grid) coevolution applied to the building of an opening book (OB) in 9x9 Go. Known sayings around the game of Go are refound by the algorithm, and the resulting program was also able to credibly comment openings in professional games of 9x9 Go. Interestingly, beyond the application to the game of Go, our algorithm can be seen as a “meta”-level for the UCT-algorithm: “UCT applied to UCT” (instead of “UCT applied to a random player” as usual), in order to build an OB. It is generic and could be applied as well for analyzing a given situation of a Markov Decision Process.

1 Introduction

The game of go is an ancient Asian game with the nice property that it is much harder for computers than chess and therefore still unsolved - whereas nowadays computers play handicap games in chess against top-level humans (the handicap is here for helping the human!), the top human Go players still win very easily handicap games against humans (with the handicap for helping the computer!). However, the handicap has strongly reduced. In 1998, a 6D human player could win with 29 handicap stones [9] against a top program, Many Faces of Go, whereas in 2008, MoGo (running on the supercomputer Huygens) could win with 9 handicap stones against a top-level human player, Kim Myungwan (8th Dan pro and winner of the US Open 2008); the human also said that MoGo could probably win also with only 8 handicap stones and was roughly 2nd or 3rd Dan. The most important tool for this strong improvement is Monte-Carlo Tree Search [12,4,8,2,7].

Complexity of OB in Go. Handcrafting OB in Go is quite hard because the usual knowledge is not encoded as a mapping *situation* → *answer* but much more as a set of local rules (termed Joseki), to be applied only after consideration of the global situation of the board. Also, a well known proverb in Go says “Learn joseki, loose two stones” (LJLTS): when a player learns these rules, he becomes weaker in a first time. The simple explanation to this fact is that local

context in Go must always be related to the global situation of the board; so, blindly applying newly learned opening sequences leads to inevitable mistakes concerning the global context of the game known as “errors in the direction of play”. Indeed, choosing between several opening sequences depending on the global context is one of the hardest skill to acquire in the game of go, because the amount of experience required to handle the emptiness of the beginning of the game is huge.

For games in which standard game algorithms (using variants of minimax) work, building automatically an OB is more or less difficult, but usually it works [1], possibly with complicated tricks for removing bad moves [5]. In the case of 9x9 Go, efficient computer players are very recent and there's no positive result on 9x9 Go OB. Also, as these recent (and strong) players are mainly best-first search, i.e. focus on a small number of moves and not on all moves, it is not clear that they can be used for building OB - OB imply some diversity. We will here show that this approach works; this result, besides its positive effectiveness, is also interesting for understanding how best-first search algorithms can be efficient for games in spite of the fact that they only study a little set of moves (without efficient expert pruning).

Best first research in games. A main element in games is the branching factor. More than the incredible number of legal situations [14], the trouble in Go is that pruning rules are quite inefficient and therefore the “practical” branching factor is not so far from the “theoretical” branching factor. This is in particular true for building OB; requesting suggestions from professional players is quite difficult when the number of possible situations is huge as in the case of Go. How do we handle this problem ? A naive approach would require 1 request for the first move, 81 for the second move, and so on. How to reduce this combinatorial explosion ? As well as MCTS is efficient in front of the combinatorial explosion of Go, we show here that a meta-level of MCTS (MCTS on top of MCTS, see below) is efficient in front of the combinatorial explosion of OB. As the detailed presentation of MCTS is beyond the scope of this paper, we here refer the reader to [4,8,2,7] for more information around MCTS, and here we insist only on the fact that MCTS is a coevolution¹.

MCTS leads to a “best-first” approach² in the sense that when we build a tree of future situations, a move is further analyzed as long as it is the move with best success rate in the simulations. Is this reasonable for OB ? The experimental answer given by this paper is positive, and we justify it as follows: (i) assume that for a fixed height d (i.e. there are d moves at most before the end), the estimate of the winning probability of winning converges to 0 or 1, depending on whether

¹ This looks obvious in our case as we have “real” games as initial points for our coevolution instead of random games, and “real” moves instead of naive random moves, but MCTS also is definitely a coevolution between variations as well.

² MCTS has often been emphasized as a compromise between exploration and exploitation, but many practitioners have seen that in the case of deterministic games, the exploration constant, when optimized, is zero or indistinguishable from zero, leading to a best-first approach.

this situation is winning or not; (ii) then for a situation with height $d + 1$, the move with best success rate will not stay the move with best success rate if it is not a winning move, and this will ensure diversification. Of course, this is quite “sequential”: no alternate move is explored if the first explored move is still at a high success rate in spite of the fact that it is not the best move. But MCTS has shown that with a strong enough computational power, this approach is in fact very reliable and better than classical approaches; we here show that it is also reliable in the difficult framework of OB in spite of the intuitive diversity requirement.

All our experiments are based on MoGo, the first program which (i) won a game against a pro player in 9x9 Go (Comp.-Go Olympiads, Amsterdam 2007), (ii) won a game against a pro player without blitz in 9x9 Go (IAGO challenge, Paris, 2008), (iii) won a game against a pro player in 19x19 Go with handicap 9 (US Go Congress, Portland, 2008). The OB built as explained below was used in the Hakone 2008 computer-Go Cup, won by MoGo. All Go figures use the PSGO package. Comments which require a high-level in 9x9 Go are given by Pierre Audouard (P.A.). P.A. is teaching Go since 1992; he was 17th in the World Amateur Go Championship (Tokyo 2008). He has a special interest in 9x9 Go and won the 1st international disabled Go tournament of 9x9 Go (Seoul, 2008). Section 2 presents the methods investigated in this paper and shows experiments in an artificial setup. Section 3 shows real-size experiments. The conclusion will emphasize some generality of the result.

2 Methods and Preliminary Results on Fast Time Settings

The goal is the incremental building of a set of games (an OB), supposed to be used by a (human or computer) player as follows: *If the current situation is in the set of games, choose the most frequent move in won games.* We choose the most simulated move in won games, instead of the move with highest success rate - there are far less erroneous moves with the most simulated move in won games. This is consistent with the MCTS literature (see e.g. [15]). We use self-play as a basic tool for building OB, as in [10] (for computer-Shogi). Precisely, while looking for a methodology for building an OB with coevolution, we understood that MCTS is precisely a tool for building OB, usually applied to a naive random player and with small time settings, and that what we need is essentially MCTS with however (i) very different time settings (ii) a good computer player; in this paper, the good player is itself the original MCTS. Our non-evolutionary methods for building OB are as follows. **Expert OB:** A handcrafted set of games played by top level players and optimized manually (even pro games can be optimized offline by experts). **4H-OB:** A version of MoGo spending 4H studying each move plays against a MoGo spending 1s per move. After 4 moves, the MoGo playing 1s per move resigns. These two MoGos use no OB and are not adaptive (no learning). The 4H-OB was rejected as weaker than the expert OB.

Coevolution has already been used for building partial strategies in Chinese chess [11] and also in the game of Go [6], but with less impressive results - our

coevolution will therefore be exactly MCTS. MCTS is usually not presented as a coevolution, and we designed our coevolutionary algorithm (CA) for OB without conceiving it initially as a MCTS; but MCTS consists in evolving a white strategy and a black strategy in a given situation, by mutating the policy in the currently studied variation. Our CA is “MCTS applied to MoGo-black and MoGo-white”, as well as MoGo is “MCTS applied to a naive random black player and a naive random white player” (not “very” naive; see [15] for details). Our co-evolutionary techniques for building OB are as follows, starting from a part of the handcrafted OB (the complete version was not available at that time).

Mutate bad moves (MBM, Algo. 1): *MoGo plays against itself (each player has 6h per side on a quad-core, roughly equivalent to 24h per side as the speed-up of MCTS for 4 cores is very good). The two MoGo use the OB and choose the move with highest success rate if at least one move with success rate > 50 % is available. Otherwise, the standard MoGo algorithm is used.*

Mutate very bad moves (MVBM): *Mogo plays against itself (each player has 6h per side). The two mogo use the OB and choose the move with highest success rate if at least one move with success rate > 10 % is available. Otherwise, the standard MoGo algorithm is used.*

MVBM was introduced because, with MBM, there were too many cases in which black did not follow any OB at all because the success rate of black was lower than 50 %. Both algorithms are a coevolution, in which an individual is a game (the games won by black (resp. white) are the black (resp. white) population; each won (resp. lost) game is a good (resp. bad) partial strategy), and as in the Parisian approach [3], the solution (the OB) is made of all the population. This algorithm is used on a grid (www.grid5000.fr, a great grid provided freely for scientific experimentation). This introduces some constraints: λ is not known in advance and not constant (λ depends on the number of available CPUs, and jobs might be preempted). In order to have preliminary results, we first run MBM on a simplified setting (see section 2.1). We see first that the depth-first approach works fine, what is a great success as building OB in Go is quite difficult; these results, quite good for white, are based on MBM. We will see however that this approach is not satisfactory for black and therefore derived the MVBM approach.

2.1 Why We Should Mutate Very Bad Moves Only

We here experiment the MBM approach, on a simplified case of 10 seconds per move, 8-cores machine; in this easier framework we could generate 3496 games. The results follow:

Conditions	Before learning	After learning	Against handcrafted OB
Success rate (white)	51.5 % \pm 1.8 %	64.3 % \pm 1.9 %	64.1 % \pm 1.8 %
Success rate (black)	48.5 % \pm 1.8 %	48.0 % \pm 1.9 %	46.1 % \pm 1.8 %

The first column sums to 100 % as it is the success rate of white (resp. black) in the case of no learning. The second column shows the results for white (resp. black) after learning against a non-learning version: the sum is higher than one as MBM did a good job on average. The third column shows that the learning has

Algorithm 1. The “mutate bad move” (MBM) algorithm. λ is the number of machines available. The random choice is performed by MoGo with long time settings (except in section 2): therefore, it is quite expensive. Readers familiar with UCT/MCTS might point out that there is no exploration term in the version of MCTS presented here; however, this is the case in many successful MCTS implementations, and other implementations often have a very small constant which is equivalent, in fact, to 0.

```

Population = small handcrafted OB.
while True do
    for  $l = 1.. \lambda$ , generate one individual (a game) in parallel with two steps as follows do
        s = initial state;  $g = (s)$ .
        while  $s$  is not a final state do
            bestScore = 0.5
            bestMove = Not A Move
            for  $m$  in the set of possible moves in  $s$  do
                score = frequency of won games in Population with move  $m$  in  $s$ 
                if  $score > bestScore$  then
                    bestScore = score
                    bestMove =  $m$ 
                end if
            end for
            if bestMove = Not A Move then
                Mutation: bestMove = RandomChoice( $s$ ).
            end if
             $s = nextState(s, bestMove)$  (transition operator)
             $g = concat(g, s)$ 
        end while
        Population = Population  $\cup \{g\}$ 
    end for
end while

```

provided better results than the handcrafted OB. However, we see no progress for black. This can be explained as follows: as long as black has not found a move with success rate $> 50\%$, he always mutates. Therefore, white improves his results by choosing moves with higher success rates, but not black. This is why in next sections, we will use the “mutate very bad moves” approach - asymptotically it is probably equivalent, but non-asymptotically it’s better to use an approach in which 40% of success rate is better than 30 % of success rate instead of repeatedly mutating in order to find at least 50 %.

2.2 Learn Joseki, Loose Two Stones (LJLTS)

These results above look quite promising, but we then tested what happens if we use this learnt OB in a stronger version of MoGo (i.e. with bigger time settings), using 60s per move instead of 10s (still on 8-cores machines).

Validation of the OB for a stronger version of MoGo. Success rate of the handcrafted OB against no OB for 60 s per move: 50 % as black, 63 % as white. Clearly the handcrafted OB still works for this stronger MoGo. Success rate of the learnt OB (learnt as above, with 10s per move games) against handcrafted OB: 30.6 % as black, 40.7 % as white. The learnt OB, learnt for 10s per move, is definitely not sufficiently strong for 60s per move. These numbers show that a weak player can’t estimate efficiently an opening sequence, whenever this weak player performs hundreds of games - the opening sequence might be good for

this weak-player, but this opening sequence will become a handicap when the player will become stronger.

Robustness of the handcrafted OB. We then switched to 300s per move, still on 8-cores machines, to see if the handcrafted OB was still efficient. Success rate of the handcrafted OB against no OB: 49.6 % as black, 57.0 % as white. We then considered an optimized handcrafted OB, still for 300s per move, and reached 49.7% as black, 62.9 % as white. This shows the LJLTS: if you learn Joseki (handcrafted opening moves) which are too difficult for you (if MoGo has only 10s per move), they are not efficient (you'd better use your own learnt opening moves). But if you become stronger (MoGo with 60s/move or 300s/move), you can trust the Joseki (the handcrafted rules).

3 Real-Size Experiments on MVBM for 9x9 Go, Komi 7.5

In this section we present the results of the real-size experiments on MVBM. Each game is played with 6h per side on a quad-core. 3000 games are played, i.e. the final population has 3000 individuals. MVBM is “Parisian”: all the population is the solution. Also, there's no removal: the population is incrementally increasing, and no individual is never eliminated (it might just become negligible). The handcrafted OB used in this section is an improved version of the one used in the previous section. We first show that the learning version becomes stronger and stronger, in front of the non-learning version. Results are presented in figure 1. The good point is that on average on the learning we have a success rate as follows against the version without OB, more than if we use the handcrafted OB (Figure 2, left). We can also check the results in self-play: is one of the player winning more and more often, converging to a resolution towards 100 % of won, i.e. a complete solving of the game ? We plot the results in figure 2 (right). High-level Go-players usually consider that with Komi 7.5, white has the advantage. This is supported by the final result (Figure 2). **Comment from P.A.:** *It is*

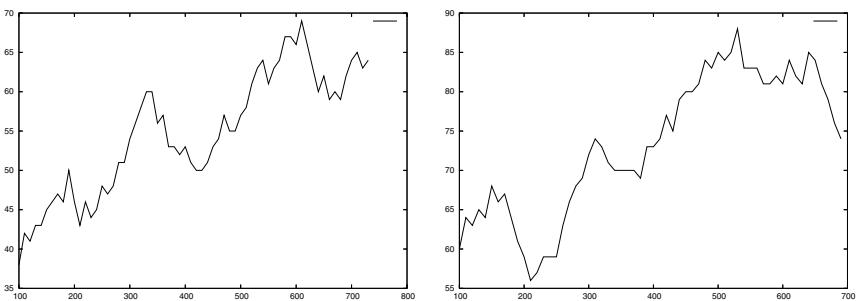


Fig. 1. Left: success rate as black against the non-learning version. Right: success rate as white against the non-learning version. For both figures, the abscissa is the time index (number of test games played); the plot is made on a moving average on 100 games.

Color	Coevolutionary opening book	Handcrafted opening book
White	74.5% \pm 2.3 %	62.9 % \pm 3.8 %
Black	64.6 % \pm 2.4 %	49.7 % \pm 3.8 %

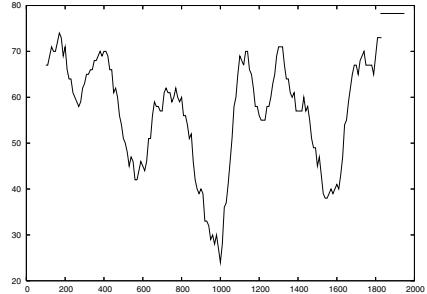


Fig. 2. Success rate of the self-learning version against itself, as white. This is consistent with usual knowledge about 9x9 Go with Komi 7.5 for strong players: the white player has the advantage.

known that with Komi 7.5 white should have the advantage on average, if the level is sufficiently high - this is therefore consistent.

Below, we present some general statistics on the optimization run and the resulting OB (section 3.1); check some known points in Go (section 3.2); present links between variations (i.e. sequences) studied by MoGo and professional games (section 3.3).

3.1 Statistics on the Final Population of Our Grid-Coevolution

We checked that all answers to all moves cited as classical in 9x9 Go in <http://senseis.xmp.net/?9x9openings> have their answer in the OB. Therefore, if a (black or white) player plays his first move in this library of moves, then an opponent playing with the OB has an automatic answer. Also, on average, the number of moves played by MoGo inside his OB against GnuGo 3.6 and Fuego 0.2.2 is 3.28 and 4.64 moves respectively.

3.2 Comments by P.A. on the OB Built by MVBM

We consider here some well known elements, proposed for test by P.A.: **Mane Go**, i.e. a strategy known very efficient for white after an initial move of black different from E5; **Kosumi**, i.e. a class of moves for which some elements are known. **Mane Go**, for the white player, consists in replying symmetrically to the black moves, for white, when black does not start with E5. With this simple strategy for white, black has to break the symmetry (by playing in the center E5) and is in a more difficult situation. P.A. tested Mane Go as follows. First move black F6: MoGo as white answers D4; then we test black D6 and MoGo as white replies symmetrically, i.e. F4. First move black F5: MoGo as white answers D5; then we test black E7 and MoGo as white replies symmetrically, i.e. E3. **Kosumi** consists in playing two stones (of the same color), in diagonal. This might be a good idea in a game, but usually not in the beginning. Here are the statistics in the final population. Many cases of Kosumi appeared just a few times and then disappeared; the only one which appeared significantly is

a Kosumi between the first and third move for black: E5 G5 F4. F4 and G5 are in Kosumi. However, this Kosumi is somewhat rare and is not chosen by MoGo when it uses the usage rule of the move used in most won games - it just appears in the exploration. However, this shows that our MVBM approach can spend time on weak moves. **Comment from P.A.:** *The absence of Kosumi for white before the seventh move is a good feature. Kosumi might happen later in the game. On the other hand, the choice E5 / G5 / F4 often studied by MoGo (though not selected at the end) for black is not a good one.* We can also check the currently analyzed openings: the last games are based on E5 / G5 / G4 / C5 or E5 / G5 / G4 / C6. This is a quite classical beginning. Also, P.A. could check that MVBM never plays as black E5/C5/E3/G5/E7 (a classical sequence of MoGo when openings are removed), which was quite bad.

3.3 Comments by MoGo on Professional Games, and Their Evaluation by a Go-Expert

Due to length constraints, this section, which can only be evaluated by 9x9-Go experts, is reported to a technical report and we only summarize here the analysis of some professional games provided by P.A. MVBM does not like the variation E5/G5/G4/C4 found in several professional games; in the population, after E5/G5/G4/C4, Black wins with probability 67.9% (whereas C5 instead of C4 is fine). According to the population, E5/G5/G4/C4/F5 is strong for black. This is consistent with pro games, as the 5 professional games with this sequence were all won by black. Also, the only game played in which black answered F4 was lost by black; this move was not considered at all by MVBM. Only one game was played with H5 (E5/G5/G4/C4/H5); according to MoGo, this sequence is very good for black; however, black lost that game. Interestingly, MVBM has simulated also the next moves until the 20th move; MVBM considers that black made a mistake later in the game and should have won with a different move (winning with probability 100% according to MVBM). **Comment by P.A.:** *MVBM is right when he says that C4 is not the best choice after E5/G5/G4 and that C5 should be chosen. Also, MVBM is right when he says that black made a mistake later in the game lost after the good initial sequence E5/G5/G4/C4/H5 - but the mistake is later than the move pointed out by MVBM. On the other hand, the move proposed by MVBM is quite strong and really leads to a good position for black.* Also, MVBM does not like the sequence E5/G5/F3; according to MVBM, White can then win with probability 75% by playing C6, or possibly C4 (70%), G3 (70%); D3 is possible also (50%) and all other moves are considered as weak by MVBM. A professional game features this sequence E5/G5/F3 with a win for black, and this is in accordance with MVBM as white then played C5, one of the bad moves according to MVBM. **Comment by P.A.:** *In spite of the fact that MVBM's prediction is in accordance with the result of the game, C5 a good move after E5/G5/F3. Other moves proposed by MoGo are also interesting and it's difficult to choose between them. C4/C6 are more complicated than C5, and G3 is a contact move, it might be dangerous - however, in 9x9, such an early contact move can be considered.* Finally, MVBM analyzed E5/G5/G4/C5. A professional player played F4 in this

situation, and MVBM explored this solution with the same preferred answer after E5/G5/G4/C5/F4 as the one played in this professional game; however, MoGo considers that F4 should be replaced by E3 with success rate 59 % instead of F4. **Comment by P.A.:** E3 is not a good idea. This is not a good reason for the result of the considered professional game.

4 Conclusion

We used a CA for generating OB for the game of Go. There's no big efficient OB available. First, the positive results follow. **First, effectiveness.** The resulting OB is satisfactory and consistent with results of professional games. It leads to quite good results in the sense that the program is much stronger with the OB than without. It is also satisfactory, in most cases, from the point of view of an expert human player: some known rules have been found independently by the algorithm (Kosumi, Mane Go, comments on professional games). **Second, grid-compliance.** Thanks to preemptable jobs (the algorithm is completely fault tolerant), the use of the grid was not too much a trouble for other users. It was also possible to have simultaneously a small number of non-preemptable jobs and plenty of preemptable jobs, so that the grid is used adaptively. A much bigger run is under progress on grid5000, and thanks to preemptable jobs has little impact on the grid. **Third, parallelization.** The speed-up is seemingly good, as (i) there's very little redundancies between individuals in the population (ii) the individuals are different just a very few moves after the initial sequence which is not randomly mutated. This is consistent with [13] which has shown that evolution strategies have a very good speed-up when the problem has a huge dimensionality: the domain here is huge [14]. **Fourth, user-friendly aspect.** It is easy to monitor the optimization, as at any fixed time step we can see which sequence is currently analyzed by the algorithm. **Fifth, generality.** We applied our algorithm to the empty Goban (initial situation), but it could be applied easily to a given situation (Tsumego analysis). Also, it can be used for analyzing a particular situation for any Markov Decision Process. After all, MVBM is an adaptive Monte-Carlo algorithm: it is based on an initial strategy, and incrementally improves this strategy, based on simulations - it is like a Monte-Carlo algorithm used for plotting a Value-At-Risk, but used adaptively. Applications in finance or power plant management is straightforward. Some less positive points are as follows: First, one of the solution proposed by the algorithm is weak according to P.A., namely E5/G5/G4/C5/E3. By the way, the solution was later replaced (in the current run), but there was no indicator, at that time, of the fact that E3 was a bad solution. We have an algorithm which is therefore consistent, but we can't guess a priori that a solution is still unstable. Also, E5/G5/F4 was not in the final result, but a long exploration has been made on this move, whereas it is, according to P.A., a poor solution that could be removed by expert knowledge around "Kosumi". Second, no free lunch: the OB generated by a poor random player (MoGo with small time settings) is useless for a better MoGo (with longer time settings). This is consistent with [5]: the presence of erroneous moves is a major trouble when building OB.

Acknowledgements. We are very grateful to B. Bouzy, T. Cazenave, A. Cohen, R. Coulom, C. Fiter, S. Gelly, B. Helmstetter, R. Munos, Z. Yu, the computer-go mailing list, the KGS community, the Cgos server, the IAGO challenge, the Recitsproque company, the French Federation of Go, and Grid5000.

References

1. Buro, M.: Toward opening book learning. *ICCA Journal* 22, 98–102 (1999)
2. Chaslot, G., Winands, M., Uiterwijk, J., van den Herik, H., Bouzy, B.: Progressive strategies for monte-carlo tree search. In: Wang, P., et al. (eds.) *Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007)*, pp. 655–661. World Scientific Publishing Co. Pte. Ltd., Singapore (2007)
3. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Polar ifs + parisian gp = efficient inverse ifs problem solving. *Genetic Programming and Evolvable Machines* 1(4), 339–361 (2000)
4. Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search. In: Ciancarini, P., van den Herik, H.J. (eds.) *Proceedings of the 5th International Conference on Computers and Games*, Turin, Italy (2006)
5. Donninger, C., Lorenz, U.: Innovative opening-book handling. In: van den Herik, H.J., Hsu, S.-C., Hsu, T.-s., Donkers, H.H.L.M.(eds.) *CG 2005. LNCS*, vol. 4250, pp. 1–10. Springer, Heidelberg (2006)
6. Drake, P., Chen, Y.-P.: Coevolving partial strategies for the game of go. In: *International Conference on Genetic and Evolutionary Methods*. CSREA Press (2008)
7. Gelly, S., Silver, D.: Combining online and offline knowledge in uct. In: *ICML 2007: Proceedings of the 24th international conference on Machine learning*, pp. 273–280. ACM Press, New York (2007)
8. Kocsis, L., Szepesvari, C.: Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) *ECML 2006. LNCS*, vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
9. Müller, M.: Computer go. *Artificial Intelligence* 134(1-2), 145–179 (2002)
10. Nagashima, J., Hashimoto, T., Iida, H.: Self-playing-based opening book tuning. *New Mathematics and Natural Computation (NMNC)* 2(02), 183–194 (2006)
11. Ong, C., Quek, H., Tan, K., Tay, A.: Discovering chinese chess strategies through coevolutionary approaches. In: *IEEE Symposium on Computational Intelligence and Games*, pp. 360–367 (2007)
12. Péret, L., Garcia, F.: On-line search for solving markov decision processes via heuristic sampling. In: de Mántaras, R.L., Saitta, L. (eds.) *ECAI*, pp. 530–534. IOS Press, Amsterdam (2004)
13. Teytaud, O., Fournier, H.: Lower bounds for evolution strategies using vc-dimension. In G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, editors, *PPSN*. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) *PPSN 2008. LNCS*, vol. 5199, pp. 102–111. Springer, Heidelberg (2008)
14. Tromp, J., Farnebäck, G.: Combinatorics of go. In: *Proceedings of 5th International Conference on Computer and Games*, Torino, Italy (May 2006)
15. Wang, Y., Gelly, S.: Modifications of UCT and sequence-like simulations for Monte-Carlo Go. In: *IEEE Symposium on Computational Intelligence and Games*, Honolulu, Hawaii, pp. 175–182 (2007)

Evolving Teams of Cooperating Agents for Real-Time Strategy Game

Paweł Lichocki¹, Krzysztof Krawiec², and Wojciech Jaśkowski²

¹ Poznań Supercomputing and Networking Center, Poznań, Poland

² Institute of Computing Science, Poznań University of Technology, Poznań, Poland
`lichocki@man.poznan.pl, {kkrawiec,wjaskowski}@cs.put.poznan.pl`

Abstract. We apply gene expression programming to evolve a player for a real-time strategy (RTS) video game. The paper describes the game, evolutionary encoding of strategies and the technical implementation of experimental framework. In the experimental part, we compare two setups that differ with respect to the used approach of task decomposition. One of the setups turns out to be able to evolve an effective strategy, while the other leads to more sophisticated yet inferior solutions. We discuss both the quantitative results and the behavioral patterns observed in the evolved strategies.

1 Introduction and Related Work

Among the genres of computer games, the popularity of real-time strategy games (RTS) increased significantly in recent years. As the acronym suggests, an important feature of an RTS game is the lack of turns: the players issue moves at an arbitrary pace. This makes timing essential, and requires the players to feature a mixture of intelligence and reflection. However, what makes RTS games distinctive is not their real-time mode of operation (which they share with, e.g., the first-person shooter games), but the strategic and tactical character. Rather than impersonating a specific character in the game, each player operates a set of *units* in a virtual world. The units are semi-autonomous: they obey commands issued by a player, but otherwise operate autonomously, carrying out some low-level tasks (like moving to a specific location, attacking an enemy unit, etc.). The game's world typically hosts also various types of resources that have to be managed. As these features are characteristic for military operations, RTS is typically considered as a subcategory of *wargames*. Contemporary iconic representatives of this genre include *Starcraft* and *Warcraft*.

Some RTS games offer software interfaces that enable substituting the human player with an encoded strategy. There are gamers who specialize in handcoding such strategies, known as ‘AIs’ or ‘bots’. Despite this fact and despite the growing interest of computational intelligence researchers in the domain of games as a whole [8,10,4,1,11], attempts to make an RTS game a playground for computational intelligence surfaced only recently [2,12]. The primary reason for the low interest of computational intelligence community in the RTS realm is probably

the inherent complexity of such games, resulting from the abundance of actors, heterogeneity of units, complexity of the environment and its dynamic character. These factors make it difficult to draw sound conclusions concerning the utility of a particular approach, parameter setting, or strategy encoding.

Trying to avoid the overwhelming complexity of off-shelf RTS games, we approach a task that is scaled-down yet preserves the essential RTS features. In particular, we verify the utility of gene expression programming as a tool for evolving a strategy for a relatively simple single-player game of gathering resources described in Section 2.

2 The Gathering Resources Game

The game of gathering resources has been originally proposed by Buro and defined within the Open Real-Time Strategy (ORTS) [3], an open-source scaled-down environment for studying algorithms for RTS games. The gathering resources game is one-person, perfect information, non-deterministic RTS game where the player controls a homogeneous set of 20 workers that operate in a discrete 2D world of 32×32 tiles (see Fig. 1), with each tile composed of 16×16 tilepoints. This results in a discrete grid of 512×512 points that may be occupied by workers, discrete resources called *minerals*, and other objects. The goal of the game is to maximize the amount of gathered minerals within the available simulation time.

The initial position of a worker is close to the *control center (base)*. The worker should approach minerals, which are grouped into several patches, mine up to 10 of them (this is worker's maximum 'capacity'), and bring the collected minerals back to the base. This task is non-trivial due to the presence of static obstacles (hills), dynamic obstacles (randomly moving 'sheep' and other workers), and the



Fig. 1. A board of the gathering resources game used in the experiments

constraint that one mineral cannot be mined by more than four workers at a time. Thus, the two elementary skills required to attain a reasonable score are obstacle avoidance and collaborative pathfinding.

The player exerts control over each worker by means of four possible actions:

- *move*(x,y) – start moving towards location (x,y) (with a predefined speed),
- *stop()* – stop moving or mining (depending on the current activity),
- *mine()* – start mining minerals (effective in the vicinity of a mineral patch),
- *drop()* – drop all minerals (effective in the vicinity of the base).

The pace of the game is determined by the speed of worker movement (4 tile-points per 1 simulation frame) and mining (1 mineral per 4 simulation frames). When played by a human, the complete game lasts for 10 minutes at 8 simulation frames per second (totalling 4800 frames), with the player allowed to issue every unit 1 command per simulation frame.

3 The Approach and Setup

3.1 Worker Control and Strategy Encoding

We implement the model of *homogenous agents*, meaning that each worker implements the same behavior, which at the top level may be viewed as a finite-state machine. To ease the emergence of well-performing solutions, we manually decompose the task into two components and encode them in separate expression trees hosted by one individual that encodes the complete strategy. The first tree, called *commander* in following, is responsible for selecting a mineral and is queried when a particular worker is free. Once a particular mineral is selected by the commander as a *target* for the worker, the control over worker is taken over by the second tree, *navigator*, which is intended to guide the worker towards the target and may refer to it via one of available terminals. The state of approaching the mineral lasts until the worker reaches it, when it switches to the 'mineral mining' state for 40 simulation frames (mining speed \times worker's capacity). After mining is over, the control over the worker is passed back to the navigator, this time substituting the base as the navigation target. When reaching the base, the worker drops the minerals and becomes ready for another round. This working cycle is fixed; only commander and navigator undergo evolution.

When designing the representation (the set of terminals and functions) we had to handle the inevitable trade-off between the compactness of representation (and the resulting cardinality of the search space) and its expression power. In order to keep the representation compact, commanders and navigators use disjoint, task-oriented data types. The former uses only scalars (real numbers) and the latter only vectors (tuples of real numbers).

Table 1 shows the set of functions and terminals used by commanders. Each time an agent has to choose which mineral to gather, the commander tree built from these components is applied to every mineral on the board to compute its 'utility'. The mineral with the highest utility is assigned to the worker; this assignment remains unchanged until the worker reaches the prescribed mineral.

Table 1. Functions and terminals (arity=0) used by the commander tree. The notation ‘ $a ? b : c$ ’ means ‘if a is true return b otherwise return c ’.

Notation	Arity	Result
SOPP(S)	1	opposite value
SINV(S)	1	inverted value
SABS(S)	1	absolute value
SSIG(S)	1	sigmoid function
SADD(S1,S2)	2	addition
SSUB(S1,S2)	2	subtraction
SMUL(S1,S2)	2	multiplication
SMAX(S1,S2)	2	maximum
SMIM(S1,S2)	2	minimum
SLEN(SR,S1,S2)	3	$SABS(SR-S1) < SABS(SR-S2) ? S1 : S2$
SIF1(SR1,SR2,S1,S2)	4	$SR1 < SR2 ? S1 : S2$
SIF2(SR1,SR2,S1,S2)	4	$SR1 >= SR2 ? S1 : S2$
S_ONE	0	1.0
S_DISTANCE	0	distance to the mineral
S_MINERAL_INDEX	0	current mineral index
S_WORKER_INDEX	0	current agent index
S_LAST_VALUE	0	previously computed value
S_AVG_VALUE	0	the average value of all minerals
S_MAX_VALUE	0	the maximum value of all minerals
S_REFERENCES	0	number of workers going to the mineral
S_VISITS	0	number of workers’ visits
S_TASKS	0	$S_REFERENCES + S_VISITS$

Table 2 presents the set of functions and the set of terminals used for encoding of navigators. In each simulation frame the ORTS simulator queries the navigator to obtain a temporary movement target for each worker. The navigator tree evaluates to a vector represented as a tuple of two real numbers: the length and the angle. The vector is then transformed into board coordinates and an appropriate $move(x,y)$ command is performed by the worker.

Both commander and navigator are encoded in a common linear chromosome using Gene Expression Programming (GEP) [5]. In canonical GEP, an expression tree is encoded as a sequence of numbers subdivided into *head* and *tail*, with the head consisting of both function and terminal symbols, and the tail containing terminal symbols only (see [6] for more details). The length of the head determines the maximal size of the expression tree. In addition to GEP’s head and tail, we introduce an extra part called *forehead* that contains only function symbols. This allows us to control the minimal number of nodes in an expression tree, which might be helpful in forcing the evolution to focus on complex solutions. Table 3 presents the complete chromosome setup.

Defining individual’s fitness as the the number of collected minerals proved ineffective in one of the preliminary runs. We noticed that evolution may stagnate

Table 2. Functions and terminals (arity=0) used by the navigator tree

Notation	Arity	Result
VOPP(V)	1	opposite vector
VINV(V)	1	vector with inverted length
VRIG(V)	1	perpendicular vector (turn counter clockwise)
VLEF(V)	1	perpendicular vector (turn clockwise)
VSHO(V)	1	normalization to a short distance
VMID(V)	1	normalization to a middle distance
VLON(V)	1	normalization to a long distance
VADD(V1,V2)	2	addition
VSUB(V1,V2)	2	subtraction
VMAX(V1,V2)	2	longer vector
VMIN(V1,V2)	2	shorter vector
VROT(VR,V)	2	rotates V towards VR
VMUL(VR,V)	2	scales V accordingly to VR
VCRD(VR,V)	2	represents V treating VR as a vector on OX axis
VIFZ(VR,V1,V2)	3	length(VR)=0 ? V1 : V2
VLEN(VR,V1,V2)	3	chooses vector with length “closer” to the VR
VDIR(VR,V1,V2)	3	chooses vector with direction “closer” to the VR
V_CLOSEST_POSITION	0	vector to the closest unit
V_CLOSEST_MOVE	0	move vector of the closest unit
V_ANTISRC	0	inverted vector to the source
V_DST	0	destination vector
V_LAST	0	vector to the previous position
V_ORDER	0	previous move vector
V_ANTIHILL	0	vector that “points” away from the closest hill patch
V_ANTIUNIT	0	vector that “points” away from the local group of units

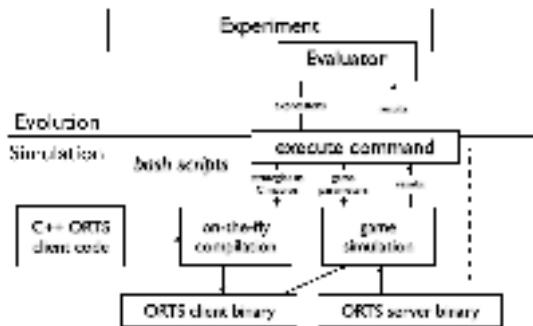
due to the discrete nature of such straightforward fitness measure. To provide an extra incentive for individuals, we defined fitness as

$$fitness = 5000 \times \left[minerals + \frac{1}{n} \sum_{i=1}^n l\left(1 - \frac{\|worker_i, target_i\|}{\|source_i, target_i\|}\right)\right], \quad (1)$$

where n is the number of workers, $source_i$ is the starting point of i^{th} worker, $target_i$ is the current target of i^{th} worker (one of the minerals or the base, depending on the current phase of the worker’s cycle), $\| \cdot \|$ stands for Euclidean distance, and $l()$ is the logistic function ($l(x) = 1/(1+\exp(-x))$). Effectively, such a definition introduces a second objective that rewards a strategy for workers’ proximity to their current targets at the end of simulation. The above formula combines these objectives in a lexicographic manner, i.e., given two strategies that collect the same number of minerals, the one that has its workers closer to their current targets is considered better. Such a definition is helpful in the initial stages of an evolutionary run, when the individuals have not discovered yet that collecting minerals is a beneficial behavior.

Table 3. Chromosome structure (numbers indicate lengths of chromosome parts)

Navigator			Commander		
Forehead (5)	Head (25)	Tail (61)	Forehead (5)	Head (25)	Tail (121)
Functions	Func+term	Terminals	Functions	Func+term	Terminals

**Fig. 2.** Experiment framework scheme

It is worth mentioning that the technical realization of the experiment was challenging. We followed and modified the previously verified technical setup for performing evolutionary experiments from [7]. We used ECJ ver. 18 [9] for running the evolutionary experiment, the ORTS framework for real-time game simulations, a homebrew parser for mapping GEP chromosomes to trees, and advanced bash-scripting to "tie the loose ends". Figure 2 shows the major software components and data flow within an experiment. ECJ standard classes are responsible for the main evolutionary loop. In the evaluation phase, individual's genotype is expressed into the C code and passed to the script, which performs on-the-fly compilation of those lines along with ORTS client code (which is constant and responsible for connecting with ORTS server). When the compilation finishes, the script runs the simulation – it starts the ORTS server and than the newly build ORTS client. When the simulation finishes, the result of the game is returned to the ECJ evaluator.

3.2 Evolutionary Parameters

Both experiments discussed in the following used the same parameter settings: population size 200, 150 generations, tournament selection with tournament size 5, and simple elitism (the best-of-generation individual is unconditionally copied to the next population). Though GEP introduces many innovative genetic operators (e.g., gene recombination, RIS-transposition), for simplicity we use the standard one-point crossover applied with probability 0.9 and one-point mutation applied with probability 0.1.

We expect that due to the large numbers of functions and terminals the search space of this problem is very large. To avoid expanding it even more, we restrain from using random constants.

The rather moderate population size and number of generations is an inevitable consequence of the time-consuming evaluation process that requires simulating the entire game. Even after shortening the game to 1000 simulation frames (compared to 4800 in the original ORTS game specification) and speeding up the simulation rate to 160 frames per second (compared to default 8 in ORTS), a 2.4GHz CPU needed more than 40 minutes to evaluate just one population, so that it took 9 days to finish the computations.

4 The Results

The major objective of the experiment was to test our approach with respect to the ability of evolving a successful cooperation between the commander and the navigator. To this aim, we performed two evolutionary runs. In the first one, the *fixed-target experiment*, we have fixed the commander to always assign the same, arbitrary chosen mineral to all workers. Thus, in this scenario, only the navigator was evolved. In the second one, called the *evolving-target experiment*, evolution of both the commander and the navigator took place.

Figure 3 shows the mean fitness graph for both experiments. From the very beginning of the runs, the evolving-target approach improves the performance of the average individual at notably better rate than the fixed-target run, which seems to stagnate already around the 40th generation. However, without more insight into the evolved solutions, it is difficult to determine the underlying cause for this difference.

In following we qualitatively analyze the behavioral changes in strategies encoded by successive best-of-generation individuals from both runs. Some of the emerging behaviours prove that the commander-navigator tandem was adapting to the game characteristic.

In the **fixed-target experiment** all agents aim at gathering mineral from the same static location and only the navigator was evolved. The target mineral was

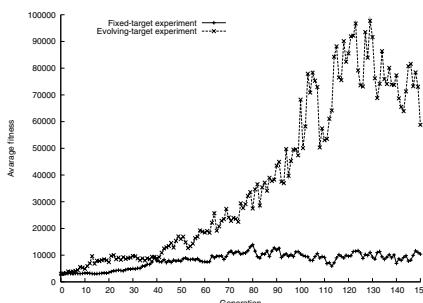
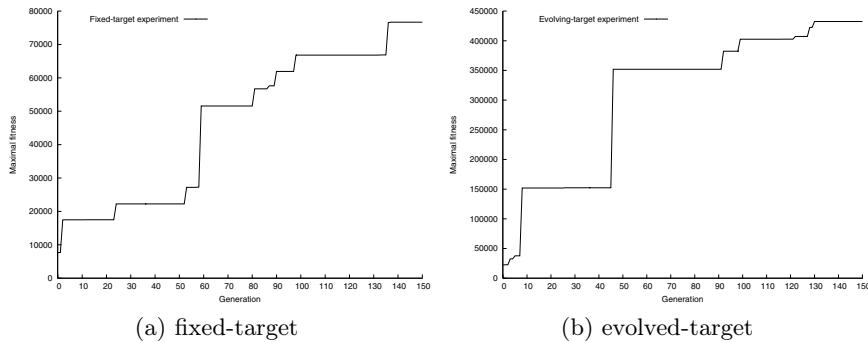


Fig. 3. Average fitness

**Fig. 4.** Best-of-generation fitness graphs

relatively far from the base and there was a wall (a static obstacle) between them. We observed qualitative changes in following generations (the best-of-generation fitness plot (Fig. 4a) reflects well these major ‘evolutionary leaps’):

(2) *Small-circle*. Agents move along small circular paths using the closest unit as the reference (the center of the circle). This is the first attempt of the agents to avoid each other (but not the hills).

(12) *Big-circle bouncing*. Agents move on large circular paths, but usually do not block each other, as two circles intersect at most only two points. Simultaneously, evolution discovers the first method of walking around the hills – agents “bounce off” the hills by rotating the move vector.

(58) *Shaking*. The previous behaviours did not prevent agents from blocking when they crowd (which inevitably happens when gathering the same mineral). In this generation agents learned to move towards a common destination by incorporating small random steps into various directions. It looks similar to Brown’s movements of chemical particles, however oriented towards a chosen point. Unfortunately, this trick does not help yet walking around the hills.

(59) *Upgraded shaking*. The agents adapted the shaking move so that it successfully helps avoiding the hills. Evolution discovered also entirely different path to reach the target mineral, which turned out to be far superior to the old one.

(137) *Shaking on demand*. When there are no other units in proximity agents tend to favor movement on straight or curved lines. “Shaking” is still used to avoid obstacles or other units.

In the **evolving-target experiment** both the navigator and the commander were evolved. The results are far better in terms of fitness, but the observed behaviors are less sophisticated. Already the initial generation turned out to contain a commander that chooses the closest mineral, and this approach dominated the remaining part of evolution (except for the 5th generation when the best individual distributes agents over different minerals).

With the commander always choosing the closest mineral, the task of the navigator becomes less sophisticated than in the fixed-target experiment, because (i) the shortest path to the target does not collide with the hills (see Fig. 1), and (ii) the short distance to be traversed does not require an elaborate mechanism for handling worker collisions. Thus, the major remaining problem to solve was the ability to move 'in herds'. Evolution discovered the *Small-circle* move in the 5th generation, the *shake* move in the 17th generation, and significantly increased the intensity of shaking in the 46th generation (Fig. 4b). The traits acquired later were minor and did not lead to any meaningful behaviour changes. Unfortunately, the agents did not learn how to walk around the base and some of them remained blocked for entire simulation.

5 Conclusions and Discussion

In general terms, the result obtained in the reported experiments may be considered as positive. Given a vocabulary of quite low-level functions and terminals, and with a little help in terms of experiment design (augmented fitness definition and task decomposition), our approach was able to evolve effective strategies that exhibit most of the functionalities required to solve the task. Let us emphasize that, apart from the assignment of workers to minerals and the ability of perceiving the board state, the evolved strategies are stateless – in a short run, the agents work exclusively using the stimulus-reaction principle. There is no short-term memory to help agents in carrying out such actions like, e.g., obstacle avoidance.

We have to admit that, considering the behavioral aspect, the evolving-target experiment produced rather disappointing outcome. Our model of artificial evolution proved very effective at cutting off the corners and, at the same time, rather ineffective at going beyond an easy-to-come-up-with yet globally not optimal solution. A lesson learned is that behavioral sophistication does not always go hand in hand with the performance.

There are numerous ways in which the proposed approach could be extended or varied. For instance, the number of functions and terminals used by commanders and navigators is large (see Tables 1 and 2). On one hand, this enables the evolving trees to express complex concepts, on the other, however, lowers the *a priori* probability of picking a particular function or terminal. We look forward for methods that address this difficult trade-off.

The fitness function defined in Formula (1) aggregates two underlying objectives: the number of collected minerals and worker's distance from the assigned target. The particular form of aggregation is quite arbitrary. In theory, a more elegant way would be to approach the problem with multi-objective formulation. However, our past experience with multiobjective evolutionary computation indicates that it inevitably implies weaker selective pressure and may negatively impact the convergence of the evolutionary run. In the light of very expensive evaluation cost in our study, such approach seemed risky. For the same reason of high computational demand, it seems reasonable to consider distributed

computing. This would also allow us to run the experiments several times to obtain a more reliable estimates of its performance.

Finally, this large-scale study could be possibly used as a yardstick for comparing GEP to GP.

Acknowledgement

This research has been supported by grant # N N519 3505 33 and EC grant QoSGrid IST FP6 STREP 033883.

References

1. Azaria, Y., Sipper, M.: GP-gammon: Genetically programming backgammon players. *Genetic Programming and Evolvable Machines* 6(3), 283–300 (2005); Published online: 12 August 2005
2. Buro, M.: Real-time strategy games: A new AI research challenge. In: *Proceedings of the International Joint Conference on AI 2003*, Acapulco, Mexico (2003)
3. Buro, M., Furtak, T.: ORTS open real time strategy framework (2005), <http://www.cs.ualberta.ca/~mburo/orts/>
4. Corno, F., Sanchez, E., Squillero, G.: On the evolution of corewar warriors. In: *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Portland, Oregon, pp. 133–138. IEEE Press, Los Alamitos (2004)
5. Ferreira, C.: Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems* 13, 87 (2001)
6. Ferreira, C.: *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Studies in Computational Intelligence. Springer, New York (2006)
7. Lichocki, P.: Evolving players for a real-time strategy game using gene expression programming. Master's thesis, Poznan University of Technology (2008)
8. Luke, S.: Genetic programming produced competitive soccer softbot teams for Robocup 1997. In: Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., Riolo, R. (eds.) *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, pp. 214–222. Morgan Kaufmann, San Francisco (1998)
9. Luke, S.: ECJ evolutionary computation system (2002), <http://cs.gmu.edu/eclab/projects/ecj/>
10. Pollack, J.B., Blair, A.D.: Co-evolution in the successful learning of backgammon strategy. *Machine Learning* 32(3), 225–240 (1998)
11. Sipper, M.: Attaining human–competitive game playing with genetic programming. In: El Yacoubi, S., Chopard, B., Bandini, S. (eds.) *ACRI 2006. LNCS*, vol. 4173, p. 13. Springer, Heidelberg (2006)
12. Stanley, K., Bryant, B., Miikkulainen, R.: Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation* 9(6), 653–668 (2005)

Design Optimization of Radio Frequency Discrete Tuning Varactors

Luís Mendes^{1,*}, Eduardo J. Solteiro Pires², Paulo B. de Moura Oliveira²,
José A. Tenreiro Machado³, Nuno M. Fonseca Ferreira⁴, João Caldinhas Vaz^{5,*},
and Maria J. Rosário^{5,*}

¹ Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Portugal
Instituto de Telecomunicações, Portugal
lmendes@estg.ipleiria.pt

² Centro de Investigação e de Tecnologias Agro-Ambientais e Biológicas,
Universidade de Trás-os-Montes e Alto Douro, Portugal
epires@utad.pt, oliveira@utad.pt

³ Inst. Sup. de Engenharia do Porto, Instituto Politécnico do Porto, Portugal
jtm@dee.isep.ipp.pt

⁴ Inst. Sup. de Engenharia de Coimbra, Instituto Politécnico de Coimbra, Portugal
fnumog@isec.pt

⁵ Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal
Instituto de Telecomunicações, Portugal
joao.vaz@ist.utl.pt, mrosario@lx.it.pt

Abstract. This work presents a procedure to automate the design of Si-integrated radio frequency (RF) discrete tuning varactors (RFDTVs). The synthesis method, which is based on evolutionary algorithms, searches for optimum performance RF switched capacitor array circuits that fulfill the design restrictions. The design algorithm uses the ϵ -dominance concept and the maximin sorting scheme to provide a set of different solutions (circuits) well distributed along an optimal front in the parameter space (circuit size and component values). Since all the solutions present the same performance, the designer can select the circuit that is best suited to be implemented in a particular integration technology. To assess the performance of the synthesis procedure, several RFDTV circuits, provided by the algorithm, were designed and simulated using a $0.18\mu\text{m}$ CMOS technology and the Cadence Virtuoso Design Platform. The comparisons between the algorithm and circuit simulation results show that they are very close, pointing out that the proposed design procedure is a powerful design tool.

Keywords: Evolutionary algorithms, analog circuit design, automated circuit synthesis and radio frequency integrated circuits.

1 Introduction

The broad-tuned inductance-capacitance (LC) resonant circuit is one of the key circuits in the implementation of reconfigurable or adaptive RF blocks for ac-

* The authors would like to thank to Fundação para a Ciência e a Tecnologia for funding the project PTDC/EEA-ELC/64368/2006.

tual and future wireless multi-standard multi-frequency transceivers. While there are several approaches to design CMOS or BiCMOS broad-tuned LC resonant circuits, the two most used methods employ RFDTVs. One of the approaches is based on RF switched capacitors arrays (RFSCAs), which uses high quality factor capacitors and RF switches, and the other is based on RF switched varactor arrays (RFSVAs), which uses varactors as capacitive switching elements. Although both techniques allow a large capacitance tuning range with small tuning steps, the RFSCAs generally present higher quality factors than the RFSVAs. Moreover, the RFSCA design is much more flexible than the one of the RFSVA. For these reasons the RFSCAs are usually preferred.

The traditional RFSCAs design method is based on a trial-and-error approach and on several rules-of-thumb. With this procedure, the designer must repeat several times the design cycle, changing each time the RFSCA components values and sizes and the number of cells, to accomplish the desired RFSCA capacitance step and range. So, this design method is very cumbersome and time consuming since the RFSCA schematic and simulations must be redone and repeated several times, respectively. Besides that, this design procedure doesn't guarantee that the final RFSCA circuit has the maximum possible performance. To improve the performance, reliability and time-to-market (by reducing substantially the design time), an innovative method based on genetic algorithms (GAs) to automate the design of optimum performance CMOS and BiCMOS RFSCAs was developed.

In the evolutionary computation field, there are single-objective problems that can be solved in a myriad of different ways, each one characterized with similar maximum fitness values. Therefore, achieving a well-spread and non-dominated parameter front is of paramount importance to the decision maker, since he can choose from a set of optimum solutions the best suited one to be developed or implemented in the problem context.

Evolutionary algorithms use different approaches to promote the solutions diversity, such as the sharing model, crowding and maximin techniques. The sharing model, originally suggested by Goldberg and Richardson [1], is used to obtain a set of optimal solutions distributed by several optimal peaks. To promote solutions in less crowded regions of the parameters space and, therefore, *forcing* the population to be well distributed, the sharing model degrades the fitness values of similar solutions according to the distance of its closest neighbors. In this method, the number of optimum solutions under each peak is proportional to its value.

The crowding technique, introduced by De Jong [2], replaces offsprings by *similar* solutions existing in the population in order to maintain its diversity. *Similarity* is measured, like in the sharing model, by evaluating the distance between the solutions. In some variants of the crowding technique, the offsprings compete against their parents, giving the winners the chance to breed in the next population. This method tends to spread solutions over the most prominent peaks, with a number of solutions in each peak proportionally to its base size.

Maximin is used in classic multi-attribute problems [3] and in game theory [4]. The maximin strategy aims to maximize the result from a set of values that

were previously obtained, from others sets, using minimizing functions. In 2003, Balling [5] proposed a multi-objective optimization technique based on a fitness function derived from the maximin strategy [4] and Li [6] used the maximin fitness in a particle swarm multi-objective optimizer. Moreover, Pires *et al.* [7] used the maximin technique to find a well distributed non-dominated Pareto front in multi-objective optimization algorithms. The maximin strategy can be used by GAs to obtain a set of solutions well distributed along the parameter space. The maximin technique is mainly used to obtain a set of points from a continuous front instead of finding the peaks of modality functions.

The method proposed in this paper uses the maximin concept together with the ϵ -dominance [8] in order to find a front in the parameter space. This method shows much better performance than the one already reported in the scientific literature [9], which uses the sharing scheme, since it reduces substantially the solutions dispersion along the optimum front and, consequently, presenting a better defined front. This paper presents a single-objective multi-parameters evolutionary algorithm, which promotes solutions diversity in the parameters space, and applies it to automate the design of optimum performance differential RFSCAs (RFDSCAs). Section 2 describes the proposed method and section 3 presents the RFDSCA circuit and its model. Besides that, the initial restrictions and objective function for this type of RFDTVs circuits are also defined in section 3. Section 4 discusses the results carried out by the proposed algorithm. Finally, section 5 outlines the main conclusions.

2 Single-Objective Multi-Parameters Optimization Algorithm

In multi-objective problems there are several ways to find the Pareto front. One of them is based on the ϵ -dominance concept in which solutions with somewhat inferior fitness values are allowed to remain in the population as non-dominated ones. This technique is used to get a set of solutions with good spread and diversity over the multi-objective space [8].

The proposed single-objective multi-parameters optimization algorithm uses the above-mention concept and the maximin technique as the main goals to achieve good diversity in the parameters space. Initially, the algorithm divides the objective space using virtual horizontal parallel hyperplanes (straight lines or planes for a 1-D or 2-D dimensional parameter space, respectively), that are separated from each other by a ϵ -distance (see figure 1(a)). Two consecutive hyperplanes define a ϵ -rank, where all the solutions have the same preference, even if their objective values are different, as illustrated in figure 1(a). After the division of the objective space, the algorithm select the solutions, starting from the better ϵ -ranks (r_1), until a ϵ -rank is found with more solutions than the empty slots available in the new population. However, if the best ϵ -rank (rank r_1) has more solutions than the number of individuals of the new population, then the solutions of the other ϵ -ranks are not considered. In both cases, the best distributed solutions in the parameter space are selected according to the

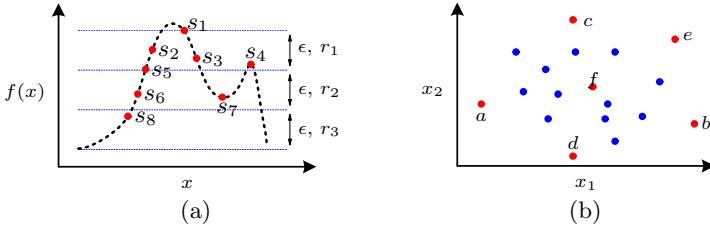


Fig. 1. (a) Problem with one objective and one parameter $\{x\}$. (b) Solutions in a bidimensional parameter space $\{x_1, x_2\}$.

maximin scheme. The main concept behind the maximin sorting scheme is to select the solutions in order to decrease the large gap areas where no solution exists in the already selected population. For example, consider the population solutions of one ϵ -rank depicted in figure 1(b). In this case, two parameters $\{x_1, x_2\}$ are considered. Initially, two extreme solutions for each parameter are selected, $\{a, b\}$ and $\{d, c\}$ for x_1 and x_2 , respectively. Through this selection the set $S \equiv \{a, b, c, d\}$ is initialized. Then, the solution e is selected because it has the larger distance to the set S . After that, solution f is selected for inclusion into the set $S \equiv \{a, b, c, d, e\}$, for the same reasons. The process is repeated until population S is completed. The maximin technique boosts its performance as the number of iterations increases, since all the solutions, which are already in the first ϵ -rank, tend to spread along a non-dominated front.

The maximin sorting scheme is depicted in figure 2. In each generation new offsprings (set D) are merged with their progenitors (set P), according with figure 2, resulting in the new set R (line 1). After that, the algorithm may select, for each one of the optimization parameters (n_{par}), the extreme solutions (getMin and getMax functions) from rank r_1 (lines 5-7) and introduces them into the final population (set S). Then, the individuals of lower rank are removed (getRankMin function) from the auxiliary population A and inserted into the set S until the number of solutions of the current rank surpass the allowed number of solutions of set S (lines 9-12). Next, the squared distance, c_{a_j} (1a), between each solution, a_j , and the solutions already selected, s_i , is evaluated (lines 13-15). After then, the solution a_j , whose squared distance to the set S is the larger (k solution), is selected (1b) (getMaxCi function, line 17). Each time a solution enters into the set S (line 18), the cost c_{a_l} of the set A is reevaluated (lines 19-21). This process ends when the set S is completed, requiring at most $O(\text{pop}_{\text{dim}}^2)$ computations, where pop_{dim} represents the number of population elements.

$$c_{a_j} = \min_{s_i \in S, a_j \in A} \| a_j - s_i \|^2 \quad (1a)$$

$$S = S \cup \{a_j : c_{a_j} = \max_{a_i \in A} c_{a_i}\} \quad (1b)$$

3 RFDSCA Automated Design Synthesis Procedure

This section presents the RFDSCA behavior model and the design steps that must be followed to obtain several RFDSCA circuits with maximum performance. The design is accomplished using the above-mentioned single-objective multi-parameters GA.

3.1 RFDSCA Behavior Model

The model used to characterize the behavior of the RFDSCAs is similar to the one reported in [10,11]. The changes introduced to the single-ended RFSCAs model were done to account for the use of a differential topology.

A generic RFDSCA circuit consists of N cells, each one constituted by two cell capacitors and a cell switch, as represented in figure 3. The first cell, named as reference cell, is comprised by two reference capacitors, both with value C , and a reference switch. This switch is formed by placing in parallel M basic switches (BS). For the other cells, the capacitors values are equal to $2^{i-1} \times C$ and the number of reference switches for each cell switch is given by 2^{i-1} , where i is the number of the cell. Besides that, the BS can be modeled by an ON resistance, R_{BS-ON} , an OFF resistance, R_{BS-OFF} , and an OFF capacitance, C_{BS-OFF} . These model components are highly dependent on the process, layout and bias.

The RFDSCA can be modeled as a non-ideal controlled capacitor, where its equivalent capacitance, C_{RFDSCA} , and quality factor, Q_{RFDSCA} , are expressed by (2) and (3), respectively. In these equations, f is the operating frequency, D represents the control word (decimal representation of the control binary word,

```

1:  $R = P \cup D$ 
2:  $S = \emptyset$ 
3:  $A = \text{getRankMin}(R)$ 
4: if  $\#A > pop_{\text{dim}}$  then
5:   for  $i = 1$  to  $n_{\text{par}}$  do
6:      $S = S \cup \text{getMin}(A, i) \cup \text{getMax}(A, i)$ 
7:   end for
8: end if
9: while  $\#S + \#A \leq pop_{\text{dim}}$  do
10:    $S = S \cup A$ 
11:    $A = \text{getRankMin}(R)$ 
12: end while
13: for  $j = 1$  to  $\#A$  do
14:    $c_{a_j} = \min_{s_i \in S} \{ \|a_j - s_i\|^2 \}$ 
15: end for
16: while  $\#S < pop_{\text{dim}}$  do
17:    $k = \text{getMaxCi}(A)$ 
18:    $S = S \cup k$ 
19:   for  $l = 1$  to  $\#A$  do
20:      $c_{a_l} = \min \{ \|a_l - k\|^2, c_{a_l} \}$ 
21:   end for
22: end while

```

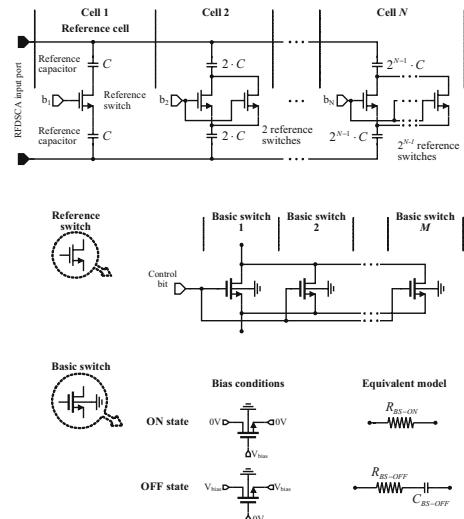


Fig. 2. Maximin pseudocode

Fig. 3. RFDSCA topology

$D = b_N \cdot 2^{N-1} + \dots + b_2 \cdot 2^1 + b_1 \cdot 2^0$) and $D_{\max} = 2^N - 1$ corresponds to the maximum value of D . The RFDSCA model is based on 6 design parameters and 2 independent variables (D and f). The design parameters that can be optimized to obtain a RFDSCA with optimum performance are N , M and C , since $R_{\text{BS-ON}}$, $R_{\text{BS-OFF}}$ and $C_{\text{BS-OFF}}$ are defined by the integration technology and bias conditions.

$$C_{\text{RFDSCA}}(D) = \frac{\frac{C}{2} + D_{\max} MC_{\text{BS-OFF}}}{1 + \frac{2MC_{\text{BS-OFF}}}{C}} \quad (2)$$

$$Q_{\text{RFDSCA}}(D, f) = \frac{\left(1 + \frac{2MC_{\text{BS-OFF}}}{C}\right) \left(1 + \frac{D}{D_{\max}} \frac{C}{2MC_{\text{BS-OFF}}}\right)}{1 + \frac{D}{D_{\max}} \left[\left(\frac{C}{2MC_{\text{BS-OFF}}} + 1\right)^2 \frac{R_{\text{BS-ON}}}{R_{\text{BS-OFF}}} - 1\right]} \quad (3)$$

3.2 RFDSCA Design Algorithm

To automate the RFDSCAs design using the algorithm described in section 2, it is necessary to define the restrictions and the fitness function appropriated for these type of circuits. The design restrictions, which are related to the initial RFDSCA specifications, are defined in equation (4).

$$C_{\text{RFDSCA-MAX}} \geq C_{\text{MAX}}, \quad C_{\text{RFDSCA-MIN}} \leq C_{\text{MIN}}, \quad \Delta C_{\text{RFDSCA}} \leq \Delta C \quad (4)$$

The three inequations show that the minimum value for the RFDSCA maximum capacitance ($C_{\text{RFDSCA-MAX}}$) must be greater or equal than the desired maximum capacitance (C_{MAX}), the maximum value for the RFDSCA minimum capacitance ($C_{\text{RFDSCA-MIN}}$) must be less or equal to the required minimum capacitance (C_{MIN}) and the maximum value for the RFDSCA capacitive tuning step (ΔC_{RFDSCA}) must be less or equal than the initially specified capacitive tuning step (ΔC). To achieve high performance RFDSCAs, it is necessary to determine the number of cells and components values and sizes that maximizes the RFDSCA quality factor. Noting that Q_{RFDSCA} decreases monotonically with f (see (3)), the objective function for this kind of RF circuits can be made independent of it. Therefore, the chosen optimization objective function is given by expression (5).

$$f_v = 2\pi \cdot f \cdot Q_{\text{RFDSCA}} \quad (5)$$

The automated design is carried out by an evolutionary algorithm. The maximin technique, that was presented in section 2, is the last operation of each iteration of the algorithm and is used to select the parent population for the next iteration. The algorithm uses 10^3 potential solutions (pop_{dim}), each one represented by the optimization circuit parameters N , M and C .

The floating point parameters are randomly initialized in an appropriate range $\{N, M, C\} = \{1\dots64, 1\dots64, 10^{-15}\dots10^{-12}\}$. The genetic evolution is then carried out over 10^5 generations. The fitness value, f_v , is given by (5) if the solution verifies the restrictions, otherwise takes a negative value, proportional to the distance to the feasible decision region, if at least one restriction of (4) is not satisfied. The successive generations of new solutions are reproduced based on a linear ranking scheme and simulated binary crossover [12]. Finally, when mutation occurs the operator replaces the value of one design parameter according to a uniform distribution function. The uniform function varies in the range $[-U, U]$ where $U = \{0.2, 0.2, 0.4 \times 10^{-15}\}$ for N , M and C , respectively. The crossover and mutation probabilities are $p_c = 0.6$ and $p_m = 0.05$, respectively.

4 RFDSCA Circuit Design and Performance Results

In this section two RFDSCAs are developed in a $0.18\mu\text{m}$ RF CMOS integration technology using the automated design synthesis procedure presented in section 3. These RFDSCAs will be used in two low power, low noise and wide tuning band CMOS LC voltage controlled oscillators, which are intended for a multi-standard multi-frequency wireless receiver. The RFDSCAs performance specifications are given in table 1(a) and the BS model components, considering the above-mentioned integration technology and the substrate parasitics, are the ones represented in table 1(b).

The optimum performance circuits obtained with the automated synthesis procedure for the two RFDSCAs specified in table 1(a) are given by the dot-points of figures 4(a) and 4(b). These two graphs present the curves of the design parameters C and N versus M for each RFDSCA, clearly showing that the algorithm finds a front with good diversity. Besides that, the algorithm convergence ability is very good since all the solutions are in the same rank. The curves of figures 4(a) and 4(b) reveal that the RFDSCAs can have several possible circuit implementations with similar performance (in other words, the problem has not a single isolated optimum peak, but, in fact an optimal front). To select a circuit with maximum quality factor for each RFDSCA, it is necessary to choose N , M and C from figures 4(a) and 4(b) that simplify the RFDSCA circuit implementation in the $0.18\mu\text{m}$ RF CMOS integration technology.

Tables 2 and 3 present several circuits for the RFDSCA_a and RFDSCA_b, respectively. These circuits were obtained using two different approaches: one based on the GA results (circuit 1 for the RFDSCA_a and circuits 3 and 4 for

Table 1. (a) Performance specifications for two RFDSCAs and (b) BS model components for the $0.18\mu\text{m}$ RF CMOS integration technology

(a)					(b)	
	C_{MIN} [fF]	C_{MAX} [fF]	ΔC [fF]	Op. Freq. [GHz]	$R_{\text{BS-ON}}$	$60\ \Omega$
RFDSCA _a	153.6	366.2	20	$3 \leq f \leq 4$	$R_{\text{BS-OFF}}$	$48\ \Omega$
RFDSCA _b	125.0	240.0	18	$5 \leq f \leq 6$	$C_{\text{BS-OFF}}$	7.24 fF

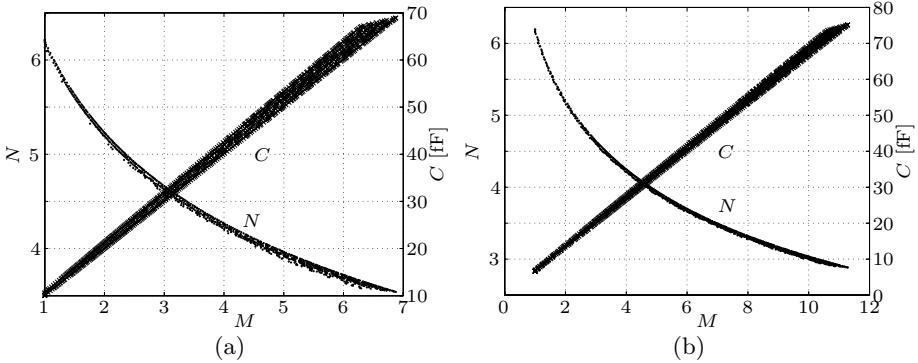


Fig. 4. Curves of the design parameters C and N versus M for the (a) RFDSCA_a and (b) RFDSCA_b

Table 2. Circuits and corresponding performances for the RFDSCA_a

	Circuit 1			Circuit 2		
	GA	Spectre	Error [%]	GA	Spectre	Error [%]
N	4.16	4	-3.85	—	4	—
M	4.07	4	-1.72	—	2	—
C	43.35	48.8	12.57	—	48.8	—
$C_{\text{RFDSCA-MAX}}$ [fF] ($\geq C_{\text{MAX}} = 240$ fF)	366.2	380	3.77	—	379	—
$C_{\text{RFDSCA-MIN}}$ [fF] ($\leq C_{\text{MAX}} = 125$ fF)	148.2	136	-8.23	—	89	—
ΔC_{RFDSCA} [fF] ($\leq \Delta C = 18$ fF)	12.9	16	24	—	20	—
$Q_{\text{RFDSCA-ON}}$ ($f = f_{\text{MAX}} = 6$ GHz)	62.2	54	-13.2	—	27	—

Table 3. Circuits and corresponding performances for the RFDSCA_b

	Circuit 3			Circuit 4			Circuit 5			Circuit 6		
	GA	Spectre	Error [%]									
N	4.18	4	-4.3	3.02	3	-0.66	—	4	—	—	3	—
M	4.08	4	-1.96	10.01	10	-0.1	—	1	—	—	4	—
C	28.3	32	13.1	67.7	68.6	1.33	—	32	—	—	68.6	—
$C_{\text{RFDSCA-MAX}}$ [fF] ($\geq C_{\text{MAX}} = 240$ fF)	240	262	9.2	240	244	1.7	—	260	—	—	244	—
$C_{\text{RFDSCA-MIN}}$ [fF] ($\leq C_{\text{MAX}} = 125$ fF)	123	119	-3.25	124.5	123	-1.2	—	56	—	—	76	—
ΔC_{RFDSCA} [fF] ($\leq \Delta C = 18$ fF)	6.83	9	31.8	16.16	17	5.2	—	13	—	—	24	—
$Q_{\text{RFDSCA-ON}}$ ($f = f_{\text{MAX}} = 6$ GHz)	64	55	-14.1	65.4	64	-2.14	—	14	—	—	26	—

RFDSCA_b) and the other using the trial-and-error design procedure referred in section 1 (circuit 2 for the RFDSCA_a and circuits 5 and 6 for RFDSCA_b). All the circuits were implemented in the Cadence Virtuoso Platform and simulated using the SpectreRF. The performance parameter shown in the two tables is the Q_{RFDSCA} at the maximum operation frequency when all the RFDSCA constitutive cells are in the ON state (when $D = D_{\max}$), which corresponds to the worst case.

The results presented in tables 2 and 3 reveal that all the circuits obtained with the GA fulfill the initial specifications and present the highest reported quality factors among all the circuits. The performance results obtained in SpectreRF for the circuits 1, 3 and 4 are similar to those given by the GA and fulfill all the design restrictions. The small percentage errors verified between the two kinds of results come from the rounding process of N and M and from the slightly differences between the values of the implemented reference capacitors and the ones given by the GA. Moreover, it can be seen that as the percentage errors of N , M and C decrease, the SpectreRF performance results of the RFDSCAs approach the ones of the GA (see the results of the circuit 4). Another interesting conclusion is that C has a major impact on the achievable Q_{RFDSCA} , since the error magnitude associated to the quality factor is similar to the one of the design parameter C . Finally, the GA results for the circuits 3 and 4 show that two completely different implementations of the same RFDSCA have almost equal performance, demonstrating the high efficiency of the RFDSCA automated synthesis procedure in the search of circuits that maximizes the RFDSCA quality factor.

Comparing the results associated to the circuits 1 and 2, for the RFDSCA_a , and 3, 4, 5 and 6, for the RFDSCA_b , it can be observed that the trial-and-error design approach can produce circuits with much lower quality factors than the ones obtained with the GA. Furthermore, the fulfilment of the design restrictions is not assured by this design procedure, as showing by the performance results of circuit 6 ($\Delta C_{\text{RFDSCA}} = 24 \text{ fF} \geq \Delta C = 18 \text{ fF}$).

The RFDSCA optimization algorithm, which was developed in C++, takes 7h30m to perform the optimization in a Pentium 4 at 3 GHz with 500 Mbytes of memory. Clearly, the time taken by the GA to find the RFDSCA optimum performance circuits will be substantially less if a computer with a recent processor and more memory is used. Even so, the time taken by the used machine to determine the optimum RFDSCA circuits is much less than the one taken by a designer. Besides that, the algorithm find multiple optimal solutions while a designer find only one, which may not be optimal.

5 Conclusions

This work proposes a procedure to automate the design of RFDSCAs. The automated design synthesis procedure is based on an evolutionary algorithm, which promotes the distribution of the design solutions along a non-dominated front in the parameters space. This method is based on the closed-form symbolic mathematical expressions of the input impedance and quality factor of the RFDSCAs. The outcome of the design procedure is a set of different circuits that fulfill

the design specifications, all having the same maximum performance. To verify the proposed synthesis method, several RFDSCAs were designed in a $0.18\mu\text{m}$ CMOS integration technology. The performance results of the RFDSCAs obtained with the GA were compared with the ones achieved by simulating them in the SpectreRF. The results comparison shows that the GA is able to reach optimal solutions regarding the optimization objective. Moreover, the GA obtains a set of solutions along the optimal front in one execution of the algorithm.

References

1. Goldberg, D.E., Richardson, J.: Genetic Algorithms with Sharing for Multi-Modal Function Optimisation. In: Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, Cambridge, USA, pp. 41–49 (1987)
2. De Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
3. Paul Yoon, K., Hwang, C.L.: Multiple Attribute Decision Making: An Introduction. Quantitative Applications in the Social Sciences. SAGE Publications, Thousand Oaks (1995)
4. Luce, R.D., Raiffa, H.: Game and Decision: Introduction and Critical Survey. John Wiley & Sons, Inc., New York (1957)
5. Balling, R.: The maximin fitness function; multi-objective city and regional planning. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 1–15. Springer, Heidelberg (2003)
6. Li, X.: Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 117–128. Springer, Heidelberg (2004)
7. Pires, E.J.S., de Moura Oliveira, P.B., Machado, J.A.T.: Multi-objective maximin sorting scheme. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 165–175. Springer, Heidelberg (2005)
8. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization. In: Proc. of the Genetic and Evolutionary Comp. Conference, pp. 439–447. Morgan Kaufmann Publishers, San Francisco (2002)
9. Mendes, L., Solteiro Pires, E.J., Vaz, J.C., Rosário, M.J.: Automated Design of Radio-Frequency Single-Ended Switched Capacitor Arrays using Genetic Algorithms. In: IEEE International Midwest Symposium on Circuits and Systems, Montréal, Canada (August 2007)
10. Mendes, L., Vaz, J.C., Rosário, M.J.: A Closed-Form Input Admittance Solution for RF and Microwave Switched Capacitor Arrays. In: IEEE International Midwest Symposium on Circuits and Systems, San Juan, Puerto Rico (August 2006)
11. Mendes, L., Vaz, J.C., Rosário, M.J.: Performance of Si-Integrated Wide-Band Single-Ended Switched Capacitor Arrays. In: 13th IEEE International Conference on Electronic Circuits and Systems, Nice, France (December 2006)
12. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Ltd., Chichester (2001)

An Evolutionary Path Planner for Multiple Robot Arms^{*}

Héctor A. Montes Venegas and J. Raymundo Marcial-Romero

Facultad de Ingeniería
Universidad Autónoma del Estado de México,
Cerro de Coatepec s/n, Toluca, Edo. de México, México
`{h.a.montes, rmarcial}@fi.uaemex.mx`
<http://fi.uaemex.mx>

Abstract. We present preliminary results of a path planner for two robotic arms sharing the same workspace. Unlike many other approaches, our planner finds collision-free paths using the robot's cartesian space as a trade-off between completeness and no workspace preprocessing. Given the high dimensionality of the search space, we use a two phase Genetic Algorithm to find a suitable path in workspaces cluttered with obstacles. Because the length of the path is unknown in advance, the planner manipulates a flexible and well crafted representation which allows the path to grow or shrink during the search process. The performance of our planner was tested on several scenarios where the only moving objects were the two robotic arms. The test scenarios force the manipulators to move through narrow spaces for which suitable and safe paths were found by the planner.

1 Introduction

Robotics is a research field that encompasses contributions from many other disciplines. In this paper we discuss results of planning collision-free motions for two robotic arms in an environment cluttered with obstacles. The problem is formulated as an optimization problem and is solved using a method based on Genetic Algorithms (GAs) [1,2].

Motion planning is an important area in robotics research. Its purpose is to automatically guide a robot around in an environment filled with obstacles. Motion planners remove the need for an operator to explicitly specify a path that a robot must follow in order to reach a desired position. Planners enable the operators to concentrate on the task, rather than determining how the manipulators should move. Scores of introductory papers and a number of books on the subject are available in the literature [3,4,5].

Motion planning comprises two distinctive problems: *path* planning and *trajectory* planning [4]. Path planning typically refers to geometric specifications

* This work was supported by the UAEMex under project grant number 2570/2007U.

of the position and orientation of robots, whereas trajectory planning also considers the linear and angular velocities involved. Therefore, path planning can be seen as a subset of trajectory planning and is usually the first step in the design of trajectories. This paper is concerned with the path planning problem for multiple manipulators.

We begin in Section 2 by covering the basics of the motion planning problem. In Section 3 the space, obstacles and robot manipulators used in our simulation are described. This is followed in Section 4 by a detailed description of our evolutionary path planner. Section 5 presents several simulated scenarios solved by our method. Section 6 concludes the paper.

2 Planning for Robot Arms

A robot arm or manipulator is a mechanical arm consisting of links and joints. Joints are either rotary or translational. In order to perform a task, a robot usually has to move from a initial *configuration* to a final one without colliding with any object present in the environment. A configuration is the set of independent parameters that define the position of any part of the robot (usually the part of interest is the tip of the arm known as *end-effector*). Given the shape of each link of a robotic arm, its configuration can be specified by the angles of its joints (see Figure 1). The number of parameters specifying the configuration of the arm is called the *degrees of freedom* (DOF).

The complexity of the motion planning problem has restricted the development of practical algorithms. Canny [6] showed that the problem is NP-complete whose complexity grows exponentially as the number of DOF increases. A large number of different approaches have been proposed over the years, each one possessing its own merits and disadvantages. Comprehensive surveys and classification of the methods developed to date can be found in publications by Hwang and Ahuja [4], J. C. Latombe [3], and Choset *et al.* [5].

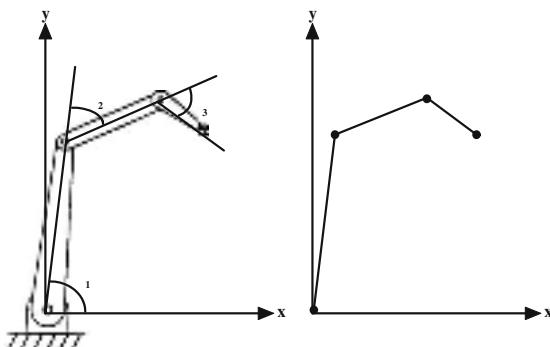


Fig. 1. The configuration parameters of a 3-link manipulator are the joint angles, θ_1 , θ_2 and θ_3 . The number of degrees of freedom is 3. The robot on the left is represented on the right as a connected sequence of line segments on a 2D cartesian space.

The method we present here is based on a GA and poses the path planning problem as an optimization problem. The goal is to minimize the end-effector's position with respect to a desired final position while the obstacle avoidance constraints are observed. The performance of our evolutionary planner is demonstrated through a series of experiments carried out on two simulated 2D manipulators each with six degrees of freedom moving among polygonal obstacles.

3 Environment Representation

Path planning is usually done involving the following steps [5]. Firstly, the configuration parameters of the robot are determined (as depicted in Figure 1). Secondly, the robots and objects need to be represented. And finally, a search method must be selected to find a solution path. We now discuss the space representation followed by the search method used by our planner.

3.1 World Space vs. Configuration Space

The world or cartesian space refers to the physical space where robots and obstacles exist. The space of all possible configurations, the configuration space (*Cspace*), of an object represents all possible motions of the object. The dimension of this space is the same as the number of DOF considered by the planner. The *Cspace* was originally introduced in motion planning in an influential paper by Lozano-Perez [7]. The fundamental idea is that by computing the *Cspace* the motion planning problem is reduced to the problem of finding a connected sequence of points between the initial and the goal configurations in the *Cspace* [4]. However, computing the *Cspace* has a major drawback as noted by the following quote from [4]:

“What makes motion planning hard is the dimensionality of the *Cspace*, i.e., the space of all possible motions of the robot. For a single rigid robot in 3D world space, the *Cspace* is 6-dimensional, and representing it with a grid requires 10^{12} points for a resolution of 100 point per dimension. Use of a grid is unrealistic for motion planning involving multi-robots.”

For this reason, the space in which safe paths are searched by our method is the cartesian space. This approach entails a trade-off between completeness and *Cspace* computing, as the *Cspace* provides a way of designing methods that guarantee a solution for any given scenario. Experimental results show that a path is found in a number of scenarios without any space pre-processing.

3.2 Object Representation

For the results reported in this paper, objects are represented using a Constructive Solid Geometry (CSG) model [4]. The CSG represents objects as unions, intersections, and set differences of primitive shapes (segment lines in our experimental scenarios). Collisions are then readily verified by examining the intersection between these shapes.

4 An Evolutionary Path Planner

Our planner is simple and shows a great potential by generating collision-free paths for multiple manipulators working in the same space. The planner uses a *subgoal network* method [4,5]. This approach finds a candidate sequence of intermediate configurations called subgoals and uses a local planner to successively move the robot through the subgoals in the sequence. If the robot cannot reach the goal configuration from any of the reached subgoals, these are stored in a list and another sequence of subgoals is found between the goal and any of the stored subgoals. The local planner is used again to check the existence of a safe path through the sequence, and the process is repeated until the goal configuration is reached. Using a local planner can be seen as a systematic way of decomposing the path planning problem into a smaller number of problems.

4.1 A Two Phase Process

We use the subgoal network method described above with an important and simple variant. Instead of finding a sequence of candidate subgoals, the planner finds only one. This subgoal has the characteristic of being as far away as possible from the stored list of subgoals previously found.

Our method comprises two steps applied in an interleaved manner. The first step, builds a representation of the accessible space by finding the next candidate subgoal, whereas the second, looks directly for the goal configuration. Both steps are formulated as optimization problems and are solved using genetic algorithms.

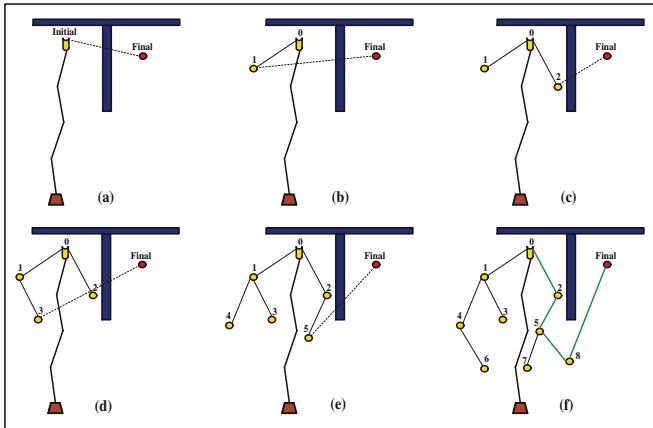


Fig. 2. The sequence (a) through (f) shows the subgoal network method used by our planner. The point labeled *Final* is the target location for the tip of the manipulator. The path to each subgoal (labeled here as 1,..,8) is found using a fixed number of motions and the one with the longest distance to the current set of goals, becomes part of that set. The final path is formed backtracking on the subgoals 8,5,2, and 0.

The first step maximizes the distance from the current candidate subgoal to the others, while the second minimizes the distance between the goal configuration and the subgoals. The method is depicted in Figure 2.

Our technique has a number of similarities with the method proposed in [8] except for two major differences; (1) our planner does not compute and search the *Cspace*, but it searches the cartesian space instead, and (2) it finds collision-free paths for multiple manipulators. A more recent evolutionary planner is presented by Smierzchalski and Michalewicz in [9]. This planner was designed as a planner/navigator for mobile robots operating in dynamic environments. Another method using evolutionary algorithms for synchronizing the trajectories of multiple robots is presented in [10]. This work uses a genetic algorithm to synchronize the trajectories independently generated for each manipulator involved in the process. Nearchou and Aspragathos [11] also present an evolutionary planner that generates paths for a single simulated robotic arm.

4.2 Path Planning for Multiple Manipulators

The path planning problem for multiple manipulators sharing the same working space have been studied for some time now [12]. Planning for multiple robots is obviously much harder since the number of DOF is the sum of the numbers of DOF of each robot. In addition, the planner must produce coordinated motion plans for all the robots involved. Several methods have been reported based on one of two methods: *centralized* and *decoupled* planning approaches [4,5]. The centralized approach consists in treating the different manipulators as one where a single planner designs the paths for all robots. One way of making this possible is to consider the cartesian product of their individual *Cspace* [3]. The decoupled approach designs the path for each separate robot in some specific order. Then the resulting paths are synchronized by either intercalating each path movement or by synchronizing the paths once they have been found. The former is called *interleaved* approach and the later *tuning* approach [5]. In this paper, we use a decentralized approach and a tuning method to synchronize the paths delivered by the planner.

4.3 Genetic Algorithms as a Search Method

Given that the two phases considered by our planner (as described in Section 4.1) can be defined as optimization problems, we use a Genetic Algorithm (GA) [2] as the search method to look for solution for both phases.

GAs are widely used to search for solutions in high-dimensional spaces like the one involving the search for collision-free paths for multiple robots with many degrees of freedom. Additionally, GAs are tolerant to the form of the function to be optimized; only a consistent evaluation of each solution is required.

Representation. The subgoal network in our method is built by generating a set of subgoals for which each path is known. This means that every subgoal is a subpath of the final path to the target destination. Each subpath is found by

first exploring the workspace in order to determine the next subgoal. First, in every generation of a GA a number of candidate subgoals are generated using a predefined number of robotic arm movements. For this paper four movements were experimentally determined to deliver an appropriate performance. The subgoal that reaches the longest distance from the current stored set of subgoals is also added to the set. Then a second GA is used to look for the target destination from the newly added subgoal. This process is repeated until the robot's end-effector is within a minimal distance of $\varepsilon = 0.01$.

Paths are encoded in the GA using a binary representation. Three elements are encoded in each gene of the chromosomes. The first is the link to be moved, the second is the direction, and the third the number of degrees to be added to the joint that causes the link displacement. Once the target destination is reached, the final path is formed by backtracking on the subgoals and joining the paths that connect the initial and the goal configuration. The search is made in the manipulators cartesian space and the set of subgoals initially contains only the initial configuration.

Fitness functions and GA control parameters. As for the GA, a tournament selection of size 5, a two-point crossover and a bit mutation operators are common operators of a classical GA and are considered sufficient for our experimental simulation. The crossover probability (CP) used for the first phase of the planner was 0.70 and the mutation probability (MP) was 0.02. For the second phase, the CP was 0.60 and the MP was 0.0333. Twenty independent runs per scenario were performed using a population size of 40 candidate solutions (i.e. paths) during 50 generations, and a collision-free path was found for all scenarios in all runs (except for the one clearly shown in Figure 3). All these values were experimentally determined.

Collision checks are carried out using a very simply procedure. Because the simplified representations of the geometry of every object used in our simulation, collisions are checked by examining the intersection between polygonal objects. Whenever such an intersection takes place, the movement that caused it is rendered invalid.

The quality of the solutions of the GA is determined using the following fitness functions. Let A be a robot arm of k DOF moving in a workspace W . The robot has n links ($n \leq k$) denoted by $L_i (i = 1, \dots, n)$. Also let B_1, B_2, \dots, B_q be the obstacles distributed in W . Similarly, let d_g and d_s be distance functions, μ be the set of current subgoals, N be the newly generated subgoal, q_i be the current end-effector's position, and q_f be the target position. Function 2 maximizes the distance from the newly added subgoal to the others subgoals, while function 1 minimizes the distance between the target configuration and the new candidate subgoal.

$$f = \min d(q_i, q_f), \text{ s.t. } L_j \cap B_i = 0 \forall_{i,j} \mid 1 \leq j \leq n, 1 \leq i \leq q \quad (1)$$

$$f = \max d(\mu, N), \text{ s.t. } L_j \cap B_i = 0 \forall_{i,j} \mid 1 \leq j \leq n, 1 \leq i \leq q \quad (2)$$

For d_s the Hausdorff metric was used to measure the distance between two sets of points, i.e., the set of current subgoals and the set formed by a new candidate subgoal. The Euclidean distance was used for d_g .

5 Experimental Results

The test environment consists of a two-dimensional cartesian space containing the simulated robots, the goal destination and various polygonal obstacles. These experimental settings are similar to environments where obstacle positions are well known in advance. These controlled environments are typical in places such as in a robotic assembly workcell [3].

We use 2 simulated 6-DOF manipulators sharing the cartesian space in which they operate. The links of the robots are presented as line segments of predefined length and the robots are the only moving objects in the environment. Each joint has a free-range motion, i.e., they can rotate 360 degrees and the collisions with other links of the same robot are depreciated. This is by no means accidental. The purpose is to examine the behaviour of the planner when bigger search spaces are explored. Finally, the dynamic properties of the manipulators are ignored and their movements are only restricted by the obstacles and by the other robot.

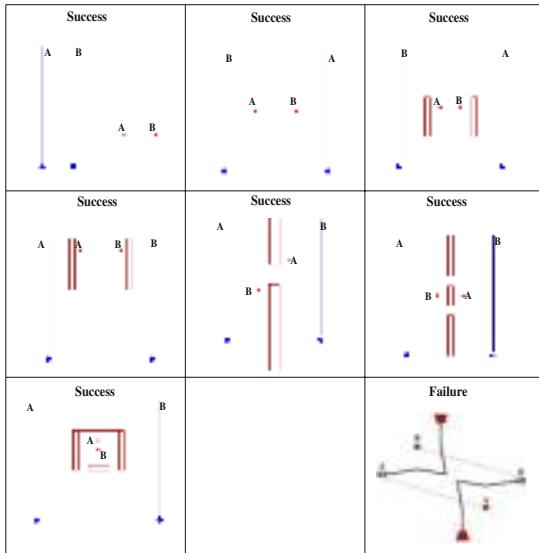


Fig. 3. A sample of eight different scenarios used to test our path planner. Each robotic arm is represented as a line of six linked segments in a full upright initial position. Each robot must reach the location labeled according to its own letter. Note that in the last scenario our planner failed to find a free collision path for both robots.

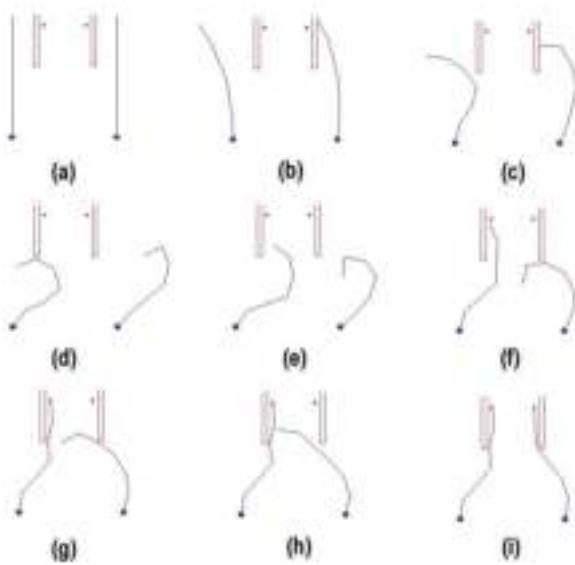


Fig. 4. Snapshots tracing the path of scenario number 4. Snapshot (a) shows both the initial and final positions for the two robots. Snapshots (b)-(i) show several positions of the robots as they follow the paths found by the planner. Similar sequences are depicted in Figures 5 and 6.

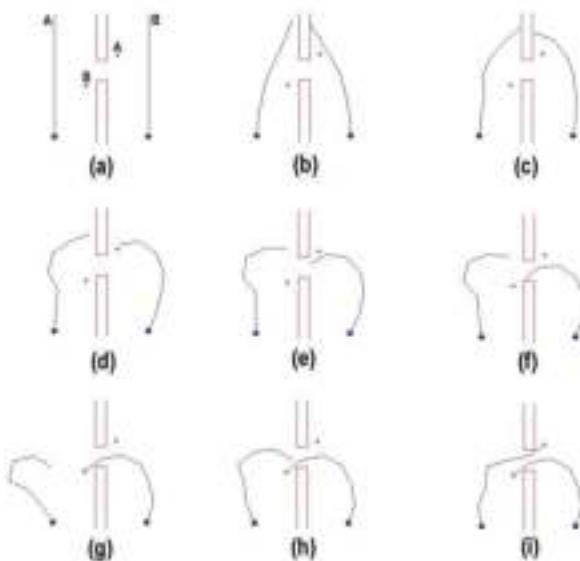


Fig. 5. Snapshots tracing the path of scenario number 5

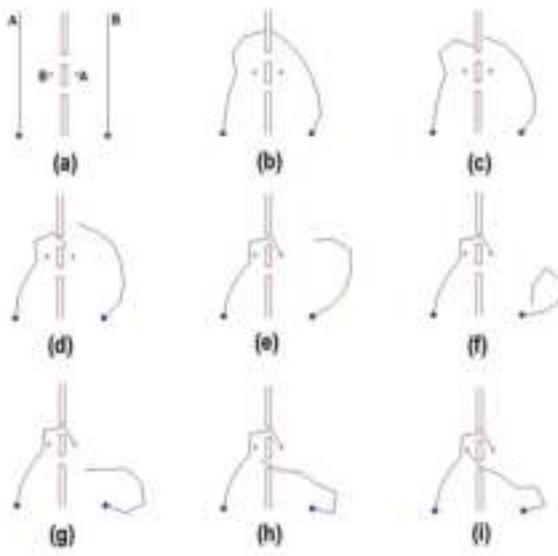


Fig. 6. Snapshots tracing the path of scenario number 6

The tip of each manipulator is iteratively guided from the initial position towards its goal using the GA to find adequate values for each robot joints. The planner was tested on several scenarios, some more elaborated than others, but all useful to demonstrate the performance of the method. Each scenario presents a deceiving environment for which other planners may perform poorly. A sample of some of the scenarios used to test the path planner is shown in Figure 3. A series of snapshots tracing the complete paths for three of these scenarios are shown in Figures 4, 5 and 6.

6 Conclusions

This paper has described preliminary results for the path planning problem for multiple manipulators operating in the same workspace. Our planner produces a plan for the robotic arms by using a strategy that combines the exploration of the free collision space while looking for the target position from each previously explored area. Both parts of this search strategy are formulated as optimization problems and solved using a genetic algorithm. Path plans are constructed using the cartesian space shared by the robot manipulators without previously computing the space of all possible configurations. The cost of this approach is the completeness of the method, for at least in one scenario the planner failed to find a solution (see Figure 3).

There are many directions in which this work may proceed and improve. One way of improving the planner is by adopting a centralized planning approach and encode the candidate paths in one single chromosome. This will reduce the

possible scenarios for which the planner is not capable of finding a solution. Another logical next step is to consider a 3D cartesian space and to use robots with geometry equivalent to those found in real life applications. Nevertheless, given the complexity of the motion planning problem, a significant amount of work is still necessary to be done before general methods can be developed.

References

1. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing, USA (1989)
2. Mitchel, M.: *An Introduction to Genetic Algorithms*. MIT Press, Boston (1998)
3. Latombe, J.C.: *Robot Motion Planning*. Kluwer Academic Publishers, Norwell (1991)
4. Hwang, Y.K., Ahuja, N.: Gross motion planning: A survey. *ACM Computing Surveys* 24(3), 219–291 (1992)
5. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: *Principles of Robot Motion: Theory, Algorithms and Implementations*. MIT Press, Boston (2005)
6. Canny, J.: *The Complexity of Robot Motion Planning*. MIT Press, Boston (1988)
7. Lozano-Perez, T.: Spatial planning: A configuration space approach. *IEEE Transactions on Computers* 32(2), 108–120 (1983)
8. Mazer, E., Ahuactzin, J.M., Bessiere, P.: The ariadne's clew algorithm. *Journal of Artificial Intelligence Research* 9, 295–316 (1998)
9. Smierzchalski, R., Michalewicz, Z.: Path planning in dynamic environments. In: *Innovations in Robot Mobility and Control. Studies in Computational Intelligence*, vol. 8, pp. 135–153. Springer, Heidelberg (2005)
10. Ridao, M.A., Riquelme, J., Camacho, E.F., Toro, M.: Automatic generation of collision-free programs for multiple manipulators using evolutive algorithms. In: *IEEE-IMACS 3rd World Multi-Conference on Circuits, Systems, Communications and Computers (CSCC 1999)*, Athenas. Computational Intelligence and Applications, pp. 239–244. World Scientific and Engineering Society Press, Singapore (1999)
11. Nearchou, A.C., Aspragathos, N.A.: A genetic path planning algorithm for redundant articulated robots. *Robotica* 15(2), 213–224 (1997)
12. Erdeman, M., Lozano-Perez, T.: On multiple moving objects. In: *Proceedings of the IEEE Conference On Robotics and Automation*, San Francisco, California, USA, pp. 1419–1424 (1986)

Evolutionary Optimization of Number of Gates in PLA Circuits Implemented in VLSI Circuits

Adam Slowik¹ and Jacek M. Zurada²

¹ Department of Electronics and Computer Science,
Koszalin University of Technology,
Sniadeckich 2 Street, 75-453 Koszalin, Poland
aslowik@ie.tu.koszalin.pl

² Department of Electrical and Computer Engineering, University of Louisville,
Louisville KY 40292, USA
j.zurada@ieee.org

Abstract. In the paper a possibility of evolutionary number of gate optimization in PLA circuits implemented in VLSI technology is presented. Multi-layer chromosomes and specialized genetic operators cooperating to them are introduced to proposed evolutionary algorithm. Due to multi-layer chromosome structures whole gates are transferred in the logic array without disturb in their structures during crossover operation. Results obtained in optimization of gate number in selection boxes of DES cryptographic algorithm are compared to results obtained using SIS program with different optimization scripts such as: rugged, algebraic, and boolean. Proposed method allows to reduce the gates number in optimized circuit. Results obtained using described evolutionary method are better than using other methods.

1 Introduction

Among elaborated combinational digital circuit design methods we can distinguish two the most popular: Karnaugh Map method, and Quine-McCluskey method. In the design of combinational digital circuits based on those methods usually only NOT, AND, and OR gate types are used. Also many different evolutionary methods [1] for design digital circuits have been developed. Among them we can mention works by Miller [2], Kalganova [3], Coello Coello [4, 5, 6], and Slowik [7, 8]. Among programs used for synthesis of logic circuits we can mention: ESPRESSO, and SIS [9]. At present, the programmable digital circuits are reaching higher and higher popularity, and among them especially PLA (*Programmable Logic Array*) circuits. These circuits consist of the AND-OR matrices which distinctive feature is that both matrices AND and OR are programmable [10]. The logical expression assigned for implementation in PLA circuits is optimized with respect to minimal number of logical products existing in this expression using Karnaugh Map method, Quine-McCluskey method, or methods of heuristics optimization available in the SIS program [9, 10]. However, this problem becomes complicated in the case when PLA circuit is to be

implemented in VLSI technology. In such a case the obtained logical expression consisted of AND, OR, and NOT gates must be transformed into logical expression which allows for circuit realization using NOR, and NOT gates. It is because both matrices (areas) AND, and OR are realized using only inverters, and NOR gates in VLSI realization. Additionally, in the case when it is necessary to compute many logical functions realized in PLA simultaneously, then this problem is more complicated, and finding the optimal solution is more difficult. In this paper, the evolutionary optimization of gate number in PLA circuits realized in VLSI technology is presented. Multi-layer chromosomes [7, 8] have been introduced into proposed algorithm, due to this chromosome structure, the whole structures of gates are moving from one to another place in the circuit during crossover operation (in single layer chromosomes the gates could be damaged during crossover). Described algorithm has been named MLCEA-PLAO (*Multi-Layer Chromosome Evolutionary Algorithm for Programmable Logic Array Optimization*), and has been tested by optimization of selection boxes (i. e. SBOX'es) in hardware realization of DES cryptographic algorithm.

2 Evolutionary Algorithm MLCEA-PLAO

2.1 Representation of Individuals

In order to create an initial population, a pattern (template) of designed circuit is created, which structure is shown in Figure 1a. The main goal of the proposed algorithm is searching for such a set of gates and connections between them in the created template, that the circuit could fulfill all constraints following from its truth table, and was composed of possible smallest number of gates. The logical expression representing a given circuit truth table has been first optimized using SIS program, and *rugged* design script, and next the technological mapping from gates set {AND, OR, NOT} to gates set {NOR, NOT} has been performed for given circuit. The gate number minimization of a given circuit (logical expression) was the main goal of technological mapping procedure. When such a solution has been obtained using the SIS program, then this solution has been introduced into the algorithm MLCEA-PLAO as an initial solution. In proposed algorithm, the multi-layer chromosomes are introduced. Its coding scheme is shown in Figure 1b (similarly as in paper [7]). Each column from 1 to t of multi-layer chromosome represents selected gate in the pattern, and rows from 1 to p represent inputs of this gate (p - number of gate inputs), row $p+1$ represent a type of the gate. In the place marked as "Input no. x" in the chromosome we put the gate number from the pattern (or the number of the circuit input), which output is to be connected to this input, in the place "Gate Type no. x" we put one of the three digits, which represent respectively: 1-NOT gate (negation of the first gate input), 2-NOR gate, 3-direct connection (DC) of the first input with the output of this gates. Digit "0" represents lack of connection of this gate input in the pattern, and digit "4" represents the circuit output. In the place "Circuit Output no. x" we put the pattern gate number, which output is to be connected to this circuit output. All circuit inputs are represented by negative numbers,

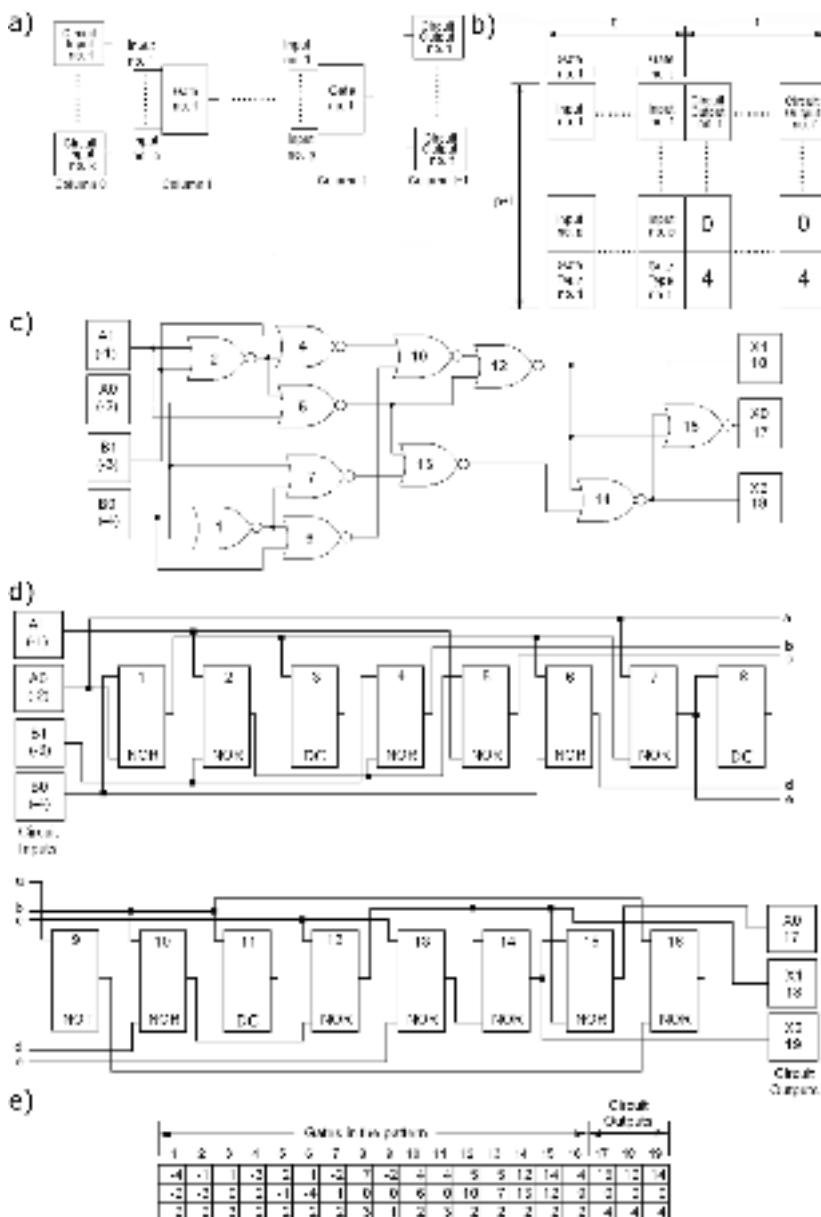


Fig. 1. Circuit coding schemes: pattern (a), multi-layer chromosome (b); exemplary circuit fulfilling the truth table for 2-bit adder (c), gate pattern representing this 2-bit adder (d), multi-layer chromosome representing gates pattern for 2-bit adder (e)

that is, the first input is represented by the number "-1", etc. In Figure 1b, first t columns of individual (multi-layer chromosome) represent successive gates in the pattern, the last n columns of the individual represent successive circuit

outputs. As an example assume a 2-bit adder having inputs A1, A0, B1, B0 and outputs X2, X1, X0. This circuit has been designed using SIS program, and *rugged* script. After the design, and circuit technological mapping to the gates set {NOT, NOR} the circuit having 16 gates ($t=16$) has been obtained. In the next step, this circuit has been introduced to the MLCEA-PLAO algorithm and optimized with respect to minimal number of its gates. In Figure 1c, the circuit obtained after stopping of proposed algorithm operation is presented; in Figure 1d, the pattern of gates for this circuit is shown, and in Figure 1e, the multi-layer chromosome representing the pattern from Figure 1d is presented.

It can be seen from Figure 1c, that obtained circuit consists of 11 gates (5 gates reduction compared to initial solution). Each of the gate has maximal number of inputs equal to 2 ($p=2$). The circuit from Figure 1c has 4 inputs: A1, A0, B1, B0, and 3 outputs: X0, X1, X2 ($n=3$). According to this, the gate structure pattern (Figure 1d), has 4 inputs, 16 places for 1-input or 2-input gates, and 3 outputs (as in initial solution circuit). The evolutionary algorithm data structure consists of multi-layer chromosome (Figure 1e) having 3 layers, and each of its layer has 19 genes ($t+n=19$).

2.2 General Concept of the Evolutionary Algorithm Operation

At the beginning we create randomly an initial circuit population avoiding feedback occurrence in the circuits (coding scheme of potential solutions - chromosomes is presented in Figure 1b). Next, the solution (circuit) obtained using SIS program and *rugged* script is inserted into initial population in the place of randomly chosen individual (chromosome). Then we evaluate a fitness according to the objective function FC for each individual:

$$fc_i = \begin{cases} v \cdot j, & \text{when } O(c_i) \neq C_i \\ 0, & \text{when } O(c_i) = C_i \end{cases} \quad (1)$$

$$FC = \begin{cases} NG, & \text{when } \sum_{i=1}^I fc_i = 0 \\ NG + w + \sum_{i=1}^I fc_i, & \text{when } \sum_{i=1}^I fc_i > 0 \end{cases} \quad (2)$$

where: v -positive real number (in experiments $v=10$), j -number of differences on particular positions in vectors $O(c_i)$ and C_i , c_i -vector containing i^{th} combination of input signals (truth table), $O(c_i)$ -vector of circuit response for vector c_i at the circuit input, C_i -vector containing proper circuit response (truth table), NG -number of gates, I -number of signal vectors in the truth table, w -penalty value (in experiments $w=10^5$). The objective function FC is minimized during the algorithm operation. After evaluation of individuals, an elitist selection is performed, and then, crossing-over and mutation (see section 2.3) are performed. Then, it is checked whether the algorithm reached accepted number of generations following from its convergence; if yes, then algorithm is stopped, and the result of algorithm operation is the circuit represented by the individual having

the lowest value of objective function FC in the population; if no the whole cycle of the algorithm is repeated.

2.3 Genetic Operators

In the paper, cross-over and mutation operators operating on $p+1$ -layer chromosomes are introduced. In the single-point cross-over operation the chromosome is cut through all $p+1$ layers in a single (randomly chosen) point; then the cut parts of two randomly selected chromosomes are exchanged. Such a cross-over of $p+1$ layer chromosomes creates new child-individuals (new circuits) with gates having unchanged input structure. In simple mutation one gene of the chromosome is chosen to mutation with equal probability. In the case when to mutation a gene from the last $p+1$ layer of chromosome is chosen, where the gate type number is written down, then an integer value from the range [1, 3] is randomly chosen, what corresponds to changing of a gate type (1-NOT, 2-NOR, 3-DC). The genes of the $p+1$ layer of the chromosome, corresponding to the columns associated with circuit outputs are not used for mutation; they always have the value "4". In the case when to mutation the gene from other layers is chosen, then one of connections between the gates in the pattern is changed.

3 Description of Experiments

During experiments the algorithm parameters were as follows: population size $ps = 100$, crossover probability $p_c = 0.35$. The value of mutation probability p_m has been determined according to formula: $p_m = \frac{1}{4(t+n)}$. The MLCEA-PLAO algorithm was stopped after 700.000 generations. It was assumed, that designed circuits are realized using 2-input gates. The MLCEA-PLAO method has been initialized using solution obtained using the SIS program, and *rugged* script. The selection boxes (*S-Box*) of DES cryptographic algorithm were taken as test circuits. Each *S-Box* (from S1 to S8) represents a converting function of 6-bit input vector into 4-bit output vector. The detailed information about truth table of particular S-Boxes can be found in [11]. In Table 1, the design results of selection boxes of DES cryptographic algorithm obtained using proposed MLCEA-PLAO method (PLAO), and obtained using SIS program and scripts: *rugged* (SIS-R), *algebraic* (SIS-A), *boolean* (SIS-B) are presented.

Table 1. Comparison of gates number included in hardware implementation of selection boxes of DES cryptographic algorithm designed using SIS program, and using MLCEA-PLAO method

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
SIS-R	217	216	189	216	186	192	199	202
SIS-A	217	222	206	215	226	248	199	238
SIS-B	214	223	208	219	202	201	197	210
PLAO	190	187	174	166	159	169	160	173

4 Conclusions

The circuits optimized using MLCEA-PLAO method are composed of lower gate number than circuits obtained using SIS program and design scripts: *rugged*, *algebraic*, *boolean*. The whole selection block of DES cryptographic algorithm consisting of 8 selection boxes has 48 inputs, and 32 outputs. When we sum up the results presented in Table 1, we can see that, to hardware realization of whole selection block: 1617 gates (SIS program and *rugged* script), 1771 gates (SIS program and *algebraic* script), 1674 (SIS program and *boolean* script), and 1378 gates (MLCEA-PLAO method) are required. It can be seen, that the circuit (whole selection block of DES cryptographic algorithm) obtained using proposed method (PLAO) has 85% of gates of circuit obtained using SIS program and script *rugged* (SIS-R), 78% of gates of circuit designed using SIS program and script *algebraic* (SIS-A), and 82% of gates of circuit generated using SIS program and script *boolean* (SIS-B).

References

1. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Heidelberg (1992)
2. Miller, J., Kalganova, T., Lipnitskaya, N., Job, D.: The Genetic Algorithm as a Discovery Engine: Strange Circuits and New Principles. In: Proceedings of the AISB Symposium on Creative Evolutionary Systems (CES 1999), Edinburgh, UK (1999)
3. Kalganova, T., Miller, J.: Evolving more efficient digital circuits by allowing circuit layout and multi-objective fitness. In: Proceedings of the First NASA/DoD Workshop on Evolvable Hardware, Los Alamitos, California, pp. 54–63 (1999)
4. Coello Coello, C.A., Christiansen, A.D., Aguirre, A.H.: Automated Design of Combinational Logic Circuits using Genetic Algorithms. In: Proc. of the Int. Conference on Artificial Neural Nets and Genetic Algorithms, pp. 335–338 (April 1997)
5. Coello Coello, C.A., Aguirre, A.H., Buckles, B.P.: Evolutionary Multiobjective Design of Combinational Logic Circuits. In: Proc. of the Second NASA/DoD Workshop on Evolvable Hardware, Los Alamitos, California, pp. 161–170 (July 2000)
6. Coello Coello, C.A., Christiansen, A.D., Aguirre, A.H.: Use of Evolutionary Techniques to Automate the Design of Combinational Circuits. International Journal of Smart Engineering System Design (2000)
7. Slowik, A., Bialko, M.: Design and Optimization of Combinational Digital Circuits Using Modified Evolutionary Algorithm. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 468–473. Springer, Heidelberg (2004)
8. Slowik, A., Bialko, M.: Evolutionary Design and Optimization of Combinational Digital Circuits with Respect to Transistor Count. Bulletin of the Polish Academy of Sciences, Technical Sciences 54(4), 437–442 (2006)
9. De Micheli, G.: *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, New York (1994)
10. Saha, A.: *Digital Principles and Logic Design*. Jones & Bartlett Publishers (2007)
11. Schneier, B.: *Applied Cryptography*. Wiley, Chichester (1996)

Particle Swarm Optimisation as a Hardware-Oriented Meta-heuristic for Image Analysis

Shahid Mehmood¹, Stefano Cagnoni²,
Monica Mordonini², and Muddassar Farooq³

¹ Center for Advanced Studies in Engineering (CASE), Islamabad, Pakistan

² Department of Information Engineering, University of Parma, Italy

³ nexGIN RC, FAST National University of Computer and Emerging Sciences
(FAST-NUCES), Islamabad, Pakistan

Abstract. In this paper we propose a variant of particle swarm optimisation (PSO), oriented at image analysis applications, that is suitable for implementation on hardware chips. The new variant, called HPSO (Hardware PSO), can be mapped easily to field-programmable gate arrays (FPGAs). The modularity of our new architecture permits to take full advantage of the active dynamic partial reconfiguration allowed by modern FPGAs. Experimental results based on simulations of a license plate detection task are presented to evaluate our design for solving real-world problems.

1 Introduction

Particle Swarm Optimisation (PSO), introduced by Kennedy and Eberhart in 1995 [1], is a meta-heuristic inspired by the behaviour of flocks of birds searching for food. In the literature, different hardware approaches for genetic [2] and Ant Colony Optimisation [3] have been reported, however only a few exists for PSO. In [4], Kókai et al. introduced a multi-swarm architecture for dynamic optimisation of adaptive array antennas in which each swarm optimises only a single parameter. The serious shortcoming of the work is that the reported results lack hardware cost, achieved frequency, and also performance. In [5], Reynolds et al. employed two Xilinx FPGAs: one for fitness calculation and the other for PSO. At 1MHz, they have reported their implementation to be approximately sixty times faster than the corresponding software one. Again they did not report the hardware cost of achieving this efficiency. Farahani et al. [6] have implemented an SOPC- based asynchronous discrete PSO. They have reported timing comparisons for three functions (OneMax, DeJong f_1 and f_2) of their SOPC-based PSO with Software PSO on Pentium IV and Nios (50MHz) processors. Moreover, they have also reported the hardware cost of SOPC-based PSO, but never studied the feasibility of their design for real-world applications.

All the work reported above implements basic PSO which is run for a fixed number of iterations, without setting any stopping condition when the desired

optimum is reached. Also, in the basic PSO, the whole swarm tends to (rapidly) converge onto a single point. In detecting non-punctual objects in images by PSO, the search space being an image, using such a *punctual fitness* function would make the search extremely sensitive to noise and possibly misleading. In fact, a meaningless isolated pixel yielding high fitness because of noise could attract and trap the whole swarm into its neighbourhood. To allow PSO to be less prone to noise and to produce a uniform distribution of particles over the target, the basic PSO algorithm needs to be modified along two directions [7]: (1) forcing division of the swarm into sub-swarms, which might converge onto different regions of interest in an image, and (2) favouring dispersion of the particles over the regions of interest in an image.

The former goal is achieved by adding a *Local_Fitness* term to the *Punctual_Fitness* to get the (overall) *Fitness* as:

$$Fitness(x, y) = Punctual_Fitness(x, y) + Local_Fitness(x, y)$$

The *Local_Fitness* depends on the number of particles with high punctual fitness within a pre-defined neighbourhood of the particle at (x, y) , and is given by:

$$Local_Fitness = K_0 \cdot number_of_neighbours$$

where K_0 is a constant.

In this way, particles tend to move towards regions where more pixels meet the punctual requirement rather than towards isolated pixels having high punctual fitness. To ensure that the swarm covers the whole extension of these regions (later goal), the basic PSO velocity-update equation is modified to include a repulsion term as:

$$v_P^*(t) = v_P(t) + repulsion_P$$

where

$$repulsion_P = \frac{\sum repulsion(i,j)}{N}$$

and

$$|repulsion(i, j)| = REPULSION_RANGE - |X_i - X_j|$$

Here i, j are particle indices and N the number of repulsive forces acting at P. Repulsive force terms are set to be zero if distance between i and j is larger than *REPULSION_RANGE*.

2 Hardware-Oriented PSO (HPSO)

HPSO for image analysis has been designed in a modular way, having FPGAs as the target architecture and implementing the fitness function as a separate entity (block). Such a modular approach is beneficial in reducing both static reconfiguration and active dynamic partial reconfiguration cost as it is simply limited to configuring the fitness function circuit.

Our PSO variant includes repulsion as well as *Local_Fitness*, as defined in section 1. In order to save area and computation time on the FPGA, some

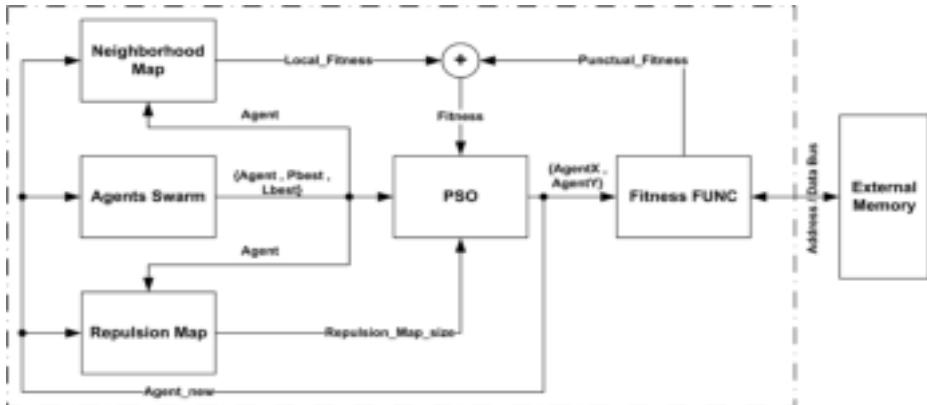


Fig. 1. HPSO Block Diagram: Blocks inside the dotted box constitute the core to be mapped on FPGA

simplifications have been made, and some constraints have been imposed on the user-defined parameters of the algorithm, such as using powers of two for the PSO constants (usually termed w , C_1 and C_2). Also image width, height and neighbourhood range within which a sub-swarm is formed are all taken to be powers of two. In our design, w is set equal to 1, while C_1 and C_2 are set equal to 2, so multiplication by such bias constants is trivially implemented by a left shift. Image width, height and neighbourhood range are set to be 1024, 512 and 32 respectively. Randomness in PSO is achieved by LFSR [8], which is shown to be sufficient for PSO needs in [5]. Figure 1 shows the block diagram of the datapath portion of the design. We now briefly describe each module.

Agents Swarm is the module where the whole swarm resides. It consists of a set of RAMs, a comparator and a Data Address Generator (DAG). The agents' RAM stores the position, velocity, and fitness of agents'. Three other RAMs (Xbest RAMs) are used to store the 'personal best' (Pbest) [1] of the agent and of its two nearest neighbours. Lbest is the best 'personal best' of such a neighbourhood. DAG generates the addresses of RAMs in this block.

Neighbourhood Map and Repulsion Map. The whole image space is divided into a number of sub-spaces of different sizes called 'maps'. This allows us to define sub-swarms of agents and to account for repulsion among swarm members. Two types of maps are considered: (1) Neighbourhood Maps, used to compute the size of the sub-swarms that are formed during the execution of the algorithm, and (2) Repulsion Maps, used to compute the amplitude of repulsion to which each particle is subjected. Within the Neighbourhood Map unit, 'actual' and 'thresholded' sub-swarm sizes are computed; the latter is used in computing *Local_Fitness*. This *Local_Fitness*, when added to the *Punctual_Fitness* from the Fitness FUNC unit, gives the (overall) *Fitness*. Repulsion Map computes repulsion as follows: whenever an agent moves, agents sharing the same Repulsion Map as the moving agent exert some repulsive force onto it. *Repulsion_Map_size* is the number of agents sharing the same starting repulsion map other than the

moving agent. This repulsive force is implemented by adding a constant amount ($\pm REPULSION_RANGE/2$) to the new velocity if *Repulsion_Map_size* is greater than threshold (1 in our case). Here repulsion carries the sign of the new velocity so that in whatever direction a particle is moving it is further repelled away.

PSO is the particle swarm optimiser (in our case, the 'object search engine') which implements the PSO equations. PSO is implemented along the two dimensions of the search space separately. PSO separability is a feature which makes the hardware implementation more attractive. In fact, our design can be easily expanded to handle n-dimensional data in future versions by simply doing a partial reconfiguration.

The Controller of the design first initialises the agents randomly in the search space and then clock iterates on the hardware, selecting randomly an agent and computing, in the PSO unit, the new agent position and velocity according to the equations reported in Section 1. The new values generated are stored in the RAMs of the Agents Swarm unit. The iterative process is stopped once the size of a sub-swarm increases above a threshold value.

3 Experimental Results

HPSO core has been synthesised and simulated using Post-Place and Route simulation model of the Xilinx ISE Design Suite. We summarize in Table 1 the results that show that the HPSO utilizes approximately 20.5% of overall available resources on Virtex II Pro XC2VP2 FPGA.

We have chosen license plate detection to test the effectiveness of our design. Since plates we considered have an aspect ratio of 5:1, we used right and left neighbourhoods of radius 2 to compute sub-swarm size. The punctual fitness of a pixel is taken to be the largest among the right and left horizontal gradients provided it belongs to a black and white region; otherwise it is taken to be zero.

We collected the statistics of results obtained on twelve car images as reported in table 2. Since PSO is a stochastic optimization technique, for a fair assessment of HPSO performance we ran the algorithm 10 times on each image, using different random seeds. A swarm of size 32 was initialised and sub-swarm size threshold for the stopping criterion was set to be 12 particles. A run was considered successful if, at convergence, the largest sub-swarm was located on the plate. Keeping this notion of being successful in mind, all 100 runs of HPSO on color car images

Table 1. Estimated Device Utilization Values for Virtex II Pro (XC2VP4-7fg456)

Logic Utilization	Used	Available	Utilization
Number of Slices	544	3008	18%
Number of Slice Flip Flops	227	6016	3%
Number of 4 input LUTs	995	6016	16%
Number of Bonded IOBs	48	248	19%
Number of BRAMs	13	28	46%
Number of MULT18X18s	6	28	21%



Fig. 2. License plate images (zoomed view) showing results of 3 successful (colour car) and 1 unsuccessful (white car) runs

Table 2. Summary of results for license plate detection task

Experiment Number	Avg. (s.dev) (iterations)	min (iterations)	max (iterations)	Avg. (clock cycles)	min (agents)	max (agents)
1	3973 (768.8)	3218	5244	21997	12	17
2	3316 (904.8)	2092	4488	18381	13	16
3	3495 (1593.5)	1840	7313	19366	12	19
4	3088 (1048.1)	1773	4712	17129	12	16
5	2351 (970.7)	1485	4257	13074	13	17
6	10910 (5151.6)	4413	19230	60150	9	17
7	2255 (835.9)	995	3789	12546	12	17
8	1378 (248.7)	974	1809	7725	12	15
9	2679 (1248.2)	1562	5836	14879	11	18
10	2527 (1032.3)	1314	5026	14043	12	14
11	1817 (661.9)	1159	3206	10139	12	17
12	7012 (3682.5)	2257	12736	38711	3	14

(Experiments 1-10) were successful. In comparison, for the black/white images (Experiments 11-12) 2 runs out of 20 were reported to be unsuccessful.

We have also noticed that, while processing images of colour cars, the largest sub-swarm was always located on the plate. However, in case of images of black/white cars, multiple sub-swarms of different sizes were formed, out of which at least one, even if not the largest one, was located over the plate. This is due to the high horizontal gradients over the car body edges and to the choice of the fitness function. This problem also hampers the sequential algorithm [7] which shares the same fitness function. However, in such an algorithm, the problem is limited by the presence of a second processing stage (Local Search), which decides about the presence of a plate at smaller sub-swarm locations.

Our revised design improved computation efficiency by 83% with respect to the version described in [9], increasing maximum operable frequency from 38.93MHz to 71.24MHz. This means that even the worst of the 120 runs (19230 iterations \sim 105911 clock cycles) took less than 1.5 ms to detect the license plate. These insights about HPSO suggest that it can be employed in tracking multiple objects simply by defining more fitness functions and activating them in turn without the need to modify the PSO block.

4 Conclusion

Our HPSO, described in this paper, can be efficiently implemented in hardware and shows good accuracy in detecting number plates in images of cars. An even more effective and efficient application of this design could be used for object tracking. Initializing the swarm close to the target reduces, rather obviously, convergence time significantly, while improving detection accuracy. Small inter-frame differences, as can be expected if a sequence is acquired at full frame rate, guarantee such a condition. Since a 'basic' detection takes at least one order of magnitude less than inter-frame arrival time (33.3 or 40 ms) in commercial video standards, further processing could be performed to improve detection robustness or extract additional information (multiple-object detection). Our future work will pursue efficient implementation of PSO and PSO-derived algorithms on both reconfigurable hardware and on unconventional architectures, such as GPUs, to produce effective and efficient embedded systems for evolutionary computer vision.

Acknowledgements

This work was supported by Higher Education Commission (HEC), Pakistan.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Int. conf. on Neural Networks, vol. IV, pp. 1942–1948 (1995)
2. Zhu, Z., Mulvaney, D., Chouliaras, V.: A novel genetic algorithm designed for hardware implementation. *Intl. J. of Computational Intelligence* 3(4), 281–288 (2006)
3. Scheuermann, B., Janson, S., Middendorf, M.: Hardware oriented ant colony optimization. *Journal of System Architecture* 53, 386–402 (2007)
4. Kókai, G., Christ, T., Frhauf, H.: Using hardware-based particle swarm method for dynamic optimization of adaptive array antennas. In: First NASA/ESA Conference on Adaptive Hardware and Systems, pp. 51–58 (2006)
5. Reynolds, P., Duren, R., Trumbo, M., Marks II, R.: FPGA implementation of particle swarm optimization for inversion of large neural networks. In: Swarm Intelligence Symposium. Proceedings 2005, pp. 389–392. Springer, Heidelberg (2005)
6. Farmahini-Farahani, A., Fakhraie, S., Safari, S.: SOPC-based architecture for discrete particle swarm optimization. In: 14th IEEE International Conference on Electronics, Circuits and Systems, pp. 1003–1006. IEEE, Los Alamitos (2007)
7. Cagnoni, S., Mordonini, M., Sartori, J.: Particle swarm optimization for object detection and segmentation. In: Giacobini, M. (ed.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 241–250. Springer, Heidelberg (2007)
8. Alfke, P.: Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators. Xilinx, Xilinx XAPP052 (1996)
9. Mehmood, S., Cagnoni, S., Mordonini, M., Matrella, G.: Hardware-oriented adaptation of a particle swarm optimization algorithm for object detection. In: 11th EUROMICRO Conference on Digital System Design, pp. 904–911. IEEE, Los Alamitos (2008)

A Novel GP Approach to Synthesize Vegetation Indices for Soil Erosion Assessment

Cesar Puente¹, Gustavo Olague¹, Stephen V. Smith¹, Stephen H. Bullock¹, Miguel A. González-Botello², and Alejandro Hinojosa-Corona¹

¹ Centro de Investigación Científica y de Educación Superior de Ensenada,
Ensenada, B.C. 22860 México

² Facultad de Ciencias. Universidad Autónoma de Baja California, Ensenada, B.C.
22800 México
{cptune,olague,svsmith}@cicese.mx

Abstract. Today the most popular method for the extraction of vegetation information from remote sensing data is through vegetation indices. In particular, erosion models are based on vegetation indices that are used to estimate the “cover factor” (C) defined by healthy, dry, or dead vegetation in a popular soil erosion model named RUSLE, (“Revised Universal Soil Loss Equation”). Several works correlate vegetation indices with C in order to characterize a broad area. However, the results are in general not suitable because most indices focus only on healthy vegetation. The aim of this study is to devise a new approach that automatically creates vegetation indices that include dry and dead plants besides healthy vegetation. For this task we propose a novel methodology based on Genetic Programming (GP) as summarized below. First, the problem is posed as a search problem where the objective is to find the index that correlates best with on field C factor data. Then, new indices are built using GP working on a set of numerical operators and bands until the best composite index is found. In this way, GP was able to develop several new indices that are better correlated compared to traditional indices such as NDVI and SAVI family. It is concluded with a real world example that it is viable to synthesize indices that are optimally correlated with the C factor using this methodology. This gives us confidence that the method could be applied in soil erosion assessment.

Keywords: Vegetation index, cover factor, genetic programming, remote sensing, soil erosion by water, RUSLE.

1 Introduction

Soil erosion is a natural process of land degradation that detaches and transports soil material through the action of an erosive agent. Possible agents are water, wind, gravity and anthropogenic activities [1]. Soil erosion by water is one of the most important land degradation problems at global scale. Erosion by water generates strong environmental impact and high economic costs because it affects directly on agricultural production and indirectly on water quality [1]. On the

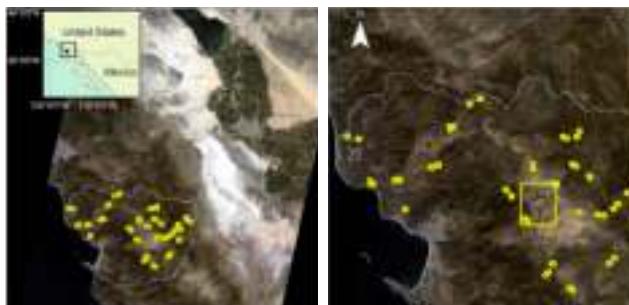


Fig. 1. Map of Baja California Mexico showing the location of Todos Santos Watershed indicating field samples used to estimate the C factor

other hand, biophysical factors that regulate soil erosion include climate (i.e., intensity of rainfall), soil type, topography, and cover. The last, cover, is one of the most crucial on reducing soil erosion. Ground cover reduces soil erosion by protecting the soil against the detachment action of falling raindrops. While the cover factor is readily measured at local plot scales, it is difficult to estimate it over broad geographic areas.

Nowadays, field surveys that estimate ground cover of a whole region are generally expensive, time consuming and labor intensive. Recently, in order to avoid such problems methods for extracting ground cover information from satellite imagery have become a powerful tool to derive biophysical properties from plants and other objects on the surface of the earth [2]. Landsat imagery are broadly used to study many aspects of our planet. The Landsat Program is a series of Earth-observing satellite missions that have collected information about Earth from space since 1972. The Thematic Mapper (TM) multi-spectral sensor was included on Landsat5 satellite. TM includes seven spectral bands covering the electromagnetic spectrum from the visible to thermal infrared [2]. For instance, band 1 whose wavelength is 450-520 nanometers (nm) represents the blue region of the spectrum. In the same way, band 2 (520-600 nm) represents green, band 3 (630-690 nm) represents red, band 4 (760-900 nm) represents near infrared, band 5 and band 7 (1550-1750 nm and 2080-2350 nm respectively) represent shortwave infrared and band 6 (10400-12500) represents thermal infrared. For simplicity, in this paper we refer these bands as *B*, *G*, *R*, *NIR*, *SWIR1*, *SWIR2* respectively. *B*, *G*, and *R* represent the visible part of the electromagnetic spectrum and *NIR*, *SWIR1*, and *SWIR2* represent the infrared spectral region. Thermal infrared is not used here.

The present work is devoted to the challenge of designing novel mathematical expressions which correspond to vegetation indices that correlate with the cover factor (C) used in erosion models. Figure 1 shows the geographic region used in our survey, where we show the location of several field samples that were acquired during the spring of 2007 and their location with respect to Landsat5-TM imagery. Thus, this paper follows a line of inquiring of previous works where vegetation indices are correlated with the cover factor [3,4,5], and in particular

we carried out a survey by sampling the Todos Santos Watershed, which covers an area of 4900 km^2 , in order to get the vegetation indices that describes a broad area of Baja California. This work is the first step towards modeling regional soil erosion caused by water using a genetic programming as a combinatorial engine that synthesizes indices that better correlate with on-site information than traditional indices from the literature. In particular, we focus on the cover factor of the RUSLE model using our GP approach. Indeed, successful results were achieved in discovering several new indices that are able to detect the combination of dry, dead, and green vegetation.

This paper is organized as follows. First, the RUSLE model is described in order to understand the role of the cover factor in the estimation of soil erosion. Second, we present some related work in order to visualize the importance of vegetation indices as a valid replacement of the cover factor. Third, we present our genetic programming methodology. Fourth, experimental results are provided to illustrate the quality of our synthetic vegetation indices. Fifth, the last section summarizes the main contributions.

2 Modeling the Cover Factor for Soil Erosion

In this section we recall the widely-used RUSLE soil erosion model and emphasize the main tendencies in measuring vegetation cover from satellite imagery. Study and prediction of land degradation processes have resulted in the creation of erosion models, such as the Morgan and Finney method, ANSWERS, WEPP and SEMMED (all cited in [5]). One of the most widely used and tested is the Universal Soil Loss Equation (USLE) [6] and its subsequent Revised Universal Soil Loss Equation (RUSLE) [7]. The USLE and RUSLE models are statistically-based water erosion models related to six erosional factors, which are defined as $A = R \times K \times L \times S \times C \times P$. Where A is the rate of soil loss measured in tons per square kilometer per year, R (in megajoule-millimeters · square kilometer · hour · year) is the rainfall-runoff factor and represent erosional energy; K (in ton-hours · megajoule · millimeter) is the soil tendency to erode; L and S are topographic factors typically combined into the LS factor. L is the slope length factor and S is the slope steepness factor; C is the vegetation cover factor; and P is the conservation support practice factor. Units for R and K determine the units for A. Other factors in RUSLE's equation are dimensionless ratios that scale field erosion estimates relative to experimental conditions under which erosion has been measured.

Cover is crucial in reducing soil erosion. As the protective foliage of cover increases; thus, soil erosion decreases. In RUSLE, factor C takes the value of 1 for soils that are constantly tilled or otherwise disturbed. These soils have the maximum potential for erosion. Another case is when soils that have not been recently disturbed, and do not have any cover (bare soil); thus, C has a nominal value of 0.45. That is, in general bare, but undisturbed soil is estimated to erode at about 45% the rate of bare, tilled soil. In the case of soils that have some cover, C is less than 0.45, depending on how sparse the vegetation is, as well as on the height of the foliage. Next, we will review previous work on vegetation indices.

2.1 Related Work about the Cover Factor

Methods for extracting C from satellite imagery have been widely used as a solution to generalize the field samples for a broad area. Traditionally, this task has been achieved through the cover classification method, which consists of assigning C values to labels that correspond to specific characteristics of the earth surface [2]. However, independently of the type of classification being applied, C cannot be estimated with high accuracy, because this approach assigns the same value to all pixels of the same label, and does not reflect the spatial variability of natural vegetation. In fact, previous work made in [8] has recognized such a drawback and it proposes a geostatistical approach to improve estimation of C in order to reduce classification errors. However, this method is expensive, and obtaining the appropriate number of samples for interpolation is difficult. Recently, a technique based on Linear Spectral Mixture Analysis (LSMA) has been proposed in [5]. LSMA is a sub-pixel classification technique, which assumes the spectrum measured by a sensor as a mixed signal that represents a linear combination of the spectra components within each pixel [9]. That work exploits the capacity of LSMA to estimate the fractional abundance of ground vegetation and bare soil simultaneously in one pixel. Hence, the classification is at sub-pixel level giving more variability to the resulting map of C. However, in order to perform an efficient un-mixing process and avoid missclassification, it is necessary to know *a priori* all the components being mixed and measured within a pixel. Moreover, because this method performs a classification, it has the same drawbacks of such methods.

From previous approaches, vegetation indices are the most used to determine C despite their drawbacks [3,4]. This use is due to the low complexity of the indices and easiness of understanding and effortless of implementation. Therefore, combinatorial possibilities not yet explored make vegetation indices a promising search space to find optimal solutions no yet proposed. In this work a genetic programming approach is chosen to exploit that search space.

2.2 Related Work about Vegetation Indices

In order to address the drawbacks from the cover classification method, vegetation indices have been explored for mapping the C factor by correlating it directly with remote sensing data by regression analysis (mainly linear regression). However, several studies report low correlation with man-made vegetation indices [3,4,5]. The reason is that the response of vegetation indices is focused only on healthy (green) vegetation. However, the condition of the vegetation is not well related to its soil protective function. For example, dead and green vegetation could reduce erosion equally effectively. A vegetation index consists in applying some mathematical analysis to the channels of satellite imagery, generally ratios or differences among bands, principal component analysis, or other linear or non-linear combinations of bands that enhance the signal of the vegetation cover presented on the surface of the earth. Despite the physical, chemical and biological theory that support data acquisition from remote sensing data [2],

the literature indicates that vegetation indices are generally designed based only on empirical evidence gained from experience of the designer [10].

The following is a brief description of how the most used vegetation indices have been developed. To know more details about these and other vegetation indices, theory that supports them, how they are designed and theirs objectives, please refer to these reviews [10,11]. A list of indices used in the literature including three evolved with our GP approach is given in Table 1.

Table 1. VI's described in this paper and their correlation coefficient ($\rho_{x,y}$) with C

Indices	Defined formulation	$\rho_{x,y}$
RV	NIR/R	-0.37
NDVI	$(NIR - R)/(NIR + R)$	-0.39
SAVI	$(1 + L) \cdot (NIR - R)/(NIR + R + L)$	-0.23
SAVI2	$(NIR)/(R + b/m)$	-0.41
TSAVI	$[m(NIR - m \cdot R - b)]/(R - m \cdot NIR - m \cdot b + X(1 + m^2))$ where m and b are the slope and intercept of the soil line. See text for details	-0.47
MSAVI	$0.5[(2NIR + 1) - \sqrt{(2NIR + 1)^2 - 8(NIR - R)}]$	-0.37
GEMI	$\eta(1 - 0.25\eta) - (R - 0.125)/(1 - R)$ where $\eta = [2(NIR^2 - R^2) + 1.5NIR + 0.5R]/(NIR + R + 0.5)$	-0.42
ARVI	$(NIR - rb)/(NIR + rb)$ where $rb = R - (B - R)$	0.38
EVI	$G[(NIR - R)/(NIR + C1 \cdot R - C2 \cdot B + L)]$ where $G = 2.5; C1 = 6; C2 = 7.5; L = 1.$	-0.38
GVI	$-0.3344B - 0.3544G - 0.4556R + 0.6966NIR + 0.0242SWIR1 - 0.2630SWIR2$	-0.64
NDWI	$(NIR - SWIR1)/(NIR + SWIR1)$	-0.40
SASI	$\beta_{SWIR1} \cdot (NIR - SWIR2)$	0.69
	Synthetic Vegetation Indices	
GPVI1	$(G - R)/[(SWIR1/SWIR2) \cdot (NIR/R) - (R/G)]$	-0.74
GPVI2	$(R \cdot SWIR2)/(\beta_{SWIR1} \cdot SWIR1)$	0.75
GPVI3	$(R - G)/SAVI2$	0.79

The first index reported was the *Ratio Vegetation Index* (RV) [10]. RV is a ratio between the near infrared (NIR) and the red (R) bands (see table 1). This ratio enhances healthy vegetation over all other objects and eliminates illumination artifacts caused by topography of the geographic area. However, its performance is dramatically low in sparse vegetation locations due to variable reflectance associated nearly bare soils. Moreover, RV ranges between 0 and infinity; this range makes it difficult to quantify the vegetation. This is the reason for the design of the *Normalized Difference Vegetation Index* (NDVI) [10], which ranges between -1 and 1.

Later, in order to approach the problem of soil sensitivity the *Soil Adjusted Vegetation Index* (SAVI) was designed [10]. In a broad geographic area, many different soil types can exist. Different soils have different reflectance spectra. RV and NDVI assume that there is a "soil line" in the R-NIR spectral space. SAVI adds a correction factor, L , that blurs this line in order to take into account the presence of different soils [10] (see table 1). Experiments show that the value for L varies between 0 for very high densities of vegetation and 1 for very low densities. The standard value typically used in most applications is 0.5, which is appropriate for intermediate vegetation densities. Many efforts have

been made to avoid the arbitrariness of L. This is the reason for the coming of the *Second SAVI* (SAVI2), *Modified SAVI* (MSAVI), *Transformed SAVI*, and (TSAVI) (see table 1), but these indices have been only partially successful [5]. The main drawback of all these indices is that they reduce soil noise at the cost of decreasing the dynamic range of the index [10]. These indices are therefore less sensitive to changes in vegetation cover than NDVI.

All indices described above also have sensitivity to atmospheric noise. The atmosphere is constantly changing and all satellite sensors have to look through it. The first attempt to deal with this problem was the *Global Environmental Monitoring Index* (GEMI) [10]. GEMI shows a complex non-linear behavior when is plotted on the R-NIR spectral space (see table 1). Authors do not give detailed reasoning for this index other than it empirically meets their requirements of insensitivity to the atmosphere [10]. Subsequent studies showed that GEMI is very sensitive to soil noise. As a result, the *Atmospherically Resistant Vegetation Indices* (ARVI) appeared. After GEMI and ARVI, an index that incorporates both soil insensitivity and atmospheric insensitivity was designed. The *Enhanced Vegetation Index* (EVI) [12] is a modified NDVI that detects vegetation and reduces the soil signal and the atmospheric influences (see table 1). Thanks to its improved performance over NDVI, EVI has gained the attention of many researchers [12].

Nearly all indices described earlier only use bands R and NIR. Atmospherically insensitive indices use band B, but despite the restriction of the indices to only a few bands, almost all satellite sensors include additional bands. There are some indices that exploit these features. The *Green Vegetation Index* (GVI) extends the concept of the soil line in the R-NIR spectral space to a soil hyperplane in the space of several bands by constructing a linear combination of these bands (see table 1). All pixels that represent soil fall in this hyperplane, and all pixels that represent vegetation fall above this hyperplane [10]. However, a different linear combination for every sensor must be calculated. The *Normalized Difference Water Index* (NDWI) [13] uses the shortwave infrared region of the spectra (SWIR bands) to detect liquid water, wet soil and plant moisture. This approach parameterizes the shape of the spectral signature by measuring the angle formed between three consecutive bands. One of these angle indices is the *Shortwave Angle Slope Index* (SASI)(see table 1) which is a combination of NIR, SWIR1 and SWIR2 of MODIS sensor bands.

3 Evolving Vegetation Indices through GP

As we have noted remote sensing imagery provides multispectral information, and in particular the R and NIR bands are widely used to design man-made vegetation indices. Thus, previous works have shown how to design vegetation indices using human expertise in order to detect vegetative cover [2,5,10,11,12,13]. However, the combinatorial possibilities of using multispectral information have not been completely explored. Therefore, we propose in this paper to use Genetic Programming (GP) as an optimization technique, in order to automatically create vegetation indices that could outperform man-made designs.

GP is a powerful machine learning technique inspired from the principles of biological evolution that is used to create computer programs that learn a target function. Today, there is an increasing emphasis on solving challenging real world applications such as the one being studied here [14]. In our work, candidate solutions being evolved by GP process, are encoded through tree-based representations that match the mathematical expressions of the vegetation indices. The trees are made up of internal and leaf nodes obtained from a set of primitive elements called function F and terminal T sets. The search space of our algorithm is defined as follows:

$$\begin{aligned} F &= \{+, -, *, /\} \\ T &= \{R, G, B, NIR, SWIR1, SWIR2, m, b, \beta_G, \beta_R, \beta_{NIR}, \beta_{SWIR1}, \\ &\quad NDVI, EVI, TSAVI, GVI, SAVI2, RVI, SASI\} \end{aligned}$$

where $R, G, B, NIR, SWIR1$, and $SWIR2$ are the image bands of the satellite Landsat5-TM as described earlier. Moreover, $\beta_G, \beta_R, \beta_{NIR}$, and β_{SWIR1} describe the angle between three consecutive bands considering the previous satellite channels. Furthermore, the m and b terminals represent the soil line parameters. Finally, to complete the terminals we consider the most used man-made vegetation indices that are represented by $NDVI, EVI, TSAVI, GVI, SAVI2, RVI$, and $SASI$. The fitness function is based on the correlation coefficient $\rho_{x,y}$ that indicates the strength and direction of the linear relationship between the factor C and the evolved vegetation indices. The correlation is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship. In this work we choose to apply the absolute value operator to $\rho_{x,y}$ because the closer the coefficient is to either -1 or 1, the stronger the correlation between the variables. Hence, the fitness function is defined as follows:

$$Q(x, y) = |\rho_{x,y}| \quad \therefore \quad \rho_{x,y} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y} = \frac{E((x - \mu_x)(y - \mu_y))}{\sigma_x \sigma_y}, \quad (1)$$

where E is the expected value operator and cov means covariance. X represents the RUSLE's C factor, y is the evolved vegetation index and $\rho_{x,y}$ is defined in the range $\{\rho_{x,y} : -1 \leq \rho_{x,y} \leq 1\}$.

4 Experimental Results

In this section we introduce the experimental results for evolving vegetation indices. The proposed approach was implemented as in general machine learning through two stages: training and testing, which were programmed in MatLab with the GP toolbox GPLAB [15]. GPLAB was tuned to perform 50 generations with a population size of 50 individuals. Moreover, initialization of the population was made by the ramped half-and-half method. The lexicographic parsimony pressure tournament was applied as selection method and the crossover and mutation probabilities were 0.70 and 0.30 respectively. Finally, elitism was applied by keeping the best individual of the parents population.

For training we used 47 data locations that were surveyed during the spring of 2007 in the Todos Santos watershed, which is located at the northwest of Mexico,

see Figure 1. These data represent the measurement of different parameters that are used to estimate the C factor, according to the methodology described in [7]. Afterwards, we identify the pixels on the Landsat5-TM imagery that correspond to each sample of the dataset. These pixels, with the corresponding band information, were then used to build the terminal set of the GP process. Finally, for testing we applied the same process using 20 new samples. During the training stage, thirty runs were made, and the best individual of each execution was kept and analyzed. Figure 2 shows runtime statistics from the best three individuals of those thirty runs. We then decide to compute a correlation matrix in order to compare our evolved indices with popular man-made vegetation indices. It can be seen that the indices found by the GP process show equal or better performance than the man-made indices (see table 1, third column).

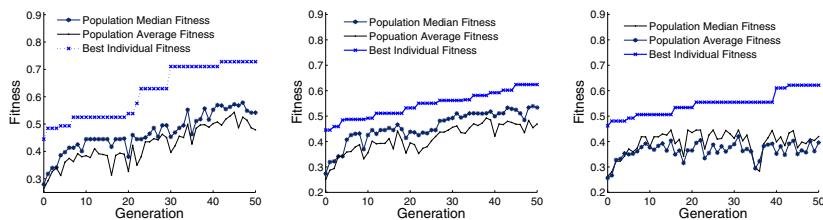


Fig. 2. Runtime statistics of the evolved indices GVPI1, GPVI2, and GVPI3 respectively

Figure 3 describes an example that illustrates the performance of the vegetation indices that have been synthesized with the GP process. The region on the original image shows several features relevant to C; for example, we can see agricultural areas, some of which were intensively cultivated at the time of the image (representing healthy vegetation); and some were not under cultivation (representing a mixture of bare soil and dry vegetation). Also a patch of land being prepared for cultivation is seen at the bottom of the image. This section represents a highly disturbed soil that corresponds to a high value for C. A part of a mountainous region is presented at the left of the image (considered to be healthy vegetation). Finally, it is easy to identify a dry riverbed that is crossing along the center of the image. This will be represented by a high value for C.

Figure 3, b), c), d), e), and f) show maps of C factor that correspond to linear regressions of NDVI, EVI, GPVI1, GPVI2, and GPVI3 respectively. As we can appreciate, each index enhances different characteristics according to the bands used for the corresponding computation. For example, all vegetation indices were able to detect the healthy vegetation areas, because all of them use information from NIR and R bands, which are basic to enhance green plants. However, it is difficult to discriminate between dry vegetation and bare soil located on abandoned agriculture areas like for NDVI and EVI. Those regions are depicted at the same grey level. By contrast, all evolved vegetation indices enhance those areas because they use information from the shortwave infrared

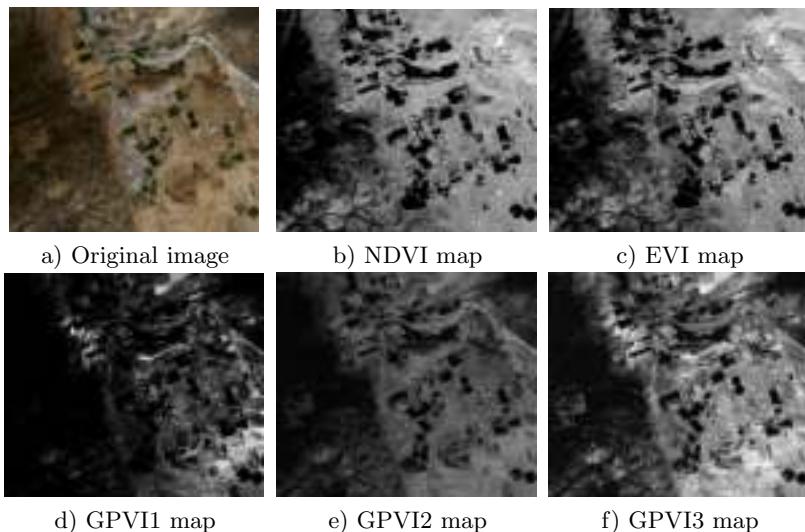


Fig. 3. Qualitative comparison between different vegetation indices. The first figure represents the original Landsat5-TM image.

bands. Therefore, the new GP vegetation indices are able to detect soil and plant moisture.

5 Conclusions and Future Work

In this work a novel genetic programming approach was described that automatically creates vegetation indices for healthy, dry and dead vegetation. In this way, several new indices were designed that result in an improved correlation coefficient computed from a real world dataset, and actually it gives better performance than widely-used indices such as NDVI and EVI. Thus, we introduce three new indices called GPVI1-3. Such indices had correlation coefficients of -0.74, 0.75, and 0.79 respectively; while, NDVI and EVI had correlation coefficients of -0.39 and -0.38. Finally, we demonstrate through a real world example that it is possible to synthesize indices that are well correlated with the C factor using the GP methodology. This gives us confidence in applying the proposed method to similar remote sensing problems.

Acknowledgements. First author would like to thank for the scholarship 179377 granted by CONACyT, México, for support his doctoral studies.

References

1. Brady, N.C., Weil, R.R.: *The Nature and Properties of Soils*, 4th edn. Pearson Prentice Hall, New Jersey (2008)
2. Ryerson, R.A. (ed.): *Manual of Remote Sensing*, 3rd edn., vol. 3. John Wiley & Sons in co-operation with the ASPRS, USA (1999)

3. de Jong, S.M.: Applications of Reflective Remote Sensing for Land Degradation Studies in a Mediterranean Environment. Universiteit Utrecht, Utrecht, Nederlands (1994)
4. Smith, S.V., Bullock, S.H., Hinojosa-Corona, A., Franco-Viscaíno, E., Escoto-Rodríguez, M., Kretzschmar, T.G., Farfan, L.M., Salazar-Cesena, J.M.: Soil Erosion and Significance for Carbon Fluxes in a Mountainous Mediterranean-Climate Watershed. *Ecol. App.* 17(5), 1379–1387 (2007)
5. Asis, A.M., Omasa, K.: Estimation of Vegetation Parameter for Modeling Soil Erosion using Linear Spectral Mixture Analysis of Landsat ETM data. *ISPRS J. of Phot. & R. S.* 62, 309–324 (2007)
6. Wischmeier, W.H., Smith, D.D.: Predicting Rainfall Erosion Losses: A Guide to Conservation Planning. Agricultural Handbook Number, vol. 537. Government Printing Office, Washington (1978)
7. Renard, K.G., Foster, G.R., Weesies, G.A., McCool, D.K., Yoder, D.C.: Predicting Soil Erosion by Water: A Guide to Conservation Planning with the Revised Universal Soil Loss Equation. Agricultural Handbook Number, vol. 703. Government Printing Office, Washington (1997)
8. Wang, G., Wente, S., Gertner, G.Z., Anderson, A.: Improvement in Mapping Vegetation Cover Factor for the Universal Soil Loss Equation by Geostatistical Methods with Landsat Thematic Mapper Images. *Int. J. of R. S.* 23(18), 3649–3667 (2002)
9. Gilabert, M.A., Garcia-Haro, F.J., Melia, J.: A mixture modeling approach to estimate vegetation parameters for heterogeneous canopies in remote sensing. *R. S. of Env.* 72(3), 328–345 (2000)
10. Ray, T.W.: A FAQ on Vegetation in Remote Sensing,
<ftp://kepler.gps.caltech.edu/pub/terrill/rsvegfaq.txt>
11. Zhao, D., Huang, L., Li, J., Qi, J.: A comparative analysis of broadband and narrowband derived vegetation indices in predicting LAI and CCD of a cotton canopy. *ISPRS J. of Phot. & R. S.* 62, 25–33 (2007)
12. Matsuchita, B., Yang, W., Chen, J., Onda, Y., Qiu, G.: Sensitivity of the Enhanced Vegetation Index (EVI) and Normalized Difference Vegetation Index (NDVI) to Topographic Effects. *Sensors* 7, 2636–2651 (2007)
13. Khana, S., Palacios-Orueta, A., Whiting, M.L., Ustin, S.L., Riaño, D., Litago, J.: Development of angle indexes for soil moisture estimation, dry matter detection and land-cover discrimination. *R. S. of Env.* 109, 154–165 (2007)
14. Poli, R., Langdon, W., McPhee, N.F.: A field guide to genetic programming. Published via and freely (2008),
<http://lulu.com>, <http://www.gp-field-guide.org.uk>
15. Silva, S.: GPLAB. A Genetic Programming Toolbox for MATLAB (2007),
<http://gplab.sourceforge.net/index.html>

Flies Open a Door to SLAM

Jean Louchet^{1,2} and Emmanuel Sapin¹

¹ INRIA-Saclay, équipe APIS, 4 rue Jacques Monod, F-91893 Orsay Cedex France

² Artenia, 24 rue Gay-Lussac, F-92320 Châtillon, France

{Jean.Louchet,Emmanuel.Sapin}@inria.fr

Abstract. The “fly algorithm” is a real-time evolutionary strategy designed for stereovision. Previous work has shown how to process stereo image sequences and use an evolving population of “flies” as a continuously updated representation of the scene for obstacle avoidance in a mobile robot, and the support to collect information about the environment from different sensors. In this paper, we move a step forward and show a way the fly representation may be used by a mobile robot for its own localisation and build a map of its environment (‘Simultaneous Localisation And Mapping’).

Keywords: Evolutionary robotics, Fly algorithm, robot vision, Parisian evolution, SLAM, image registration.

1 The Fly Algorithm

1.1 Robotics and Parisian Evolution

Robot vision is known as a computationally expensive process [9]. Contrary to the general belief that artificial evolution is too slow and complex to suit real-time applications, we showed that the Fly algorithm [10], an evolution strategy based on the Parisian evolution paradigm, is an efficient way to exploit asynchronous data delivery and fulfil the real-time constraints found in robot vision [2]. Parisian evolution essentially differs from conventional artificial evolution by its semantics. According to the general scheme of Parisian evolution [3], the problem's solution is represented by the whole population rather than the best individual alone.

A matching asynchronous robot path planner using the fly algorithm's output was proposed in [2]. In this paper, we open a way to use the flies to enable a robot locate itself and build a map of its environment.

1.2 Flies and Stereovision

A fly is defined as a 3-D point with coordinates (x, y, z) . If the fly is on the surface of an opaque object, then the corresponding pixels in the two images will normally have highly similar neighbourhoods (figure 1). Conversely, if the fly is not on the surface of an object, their close neighbourhoods will usually be poorly correlated.

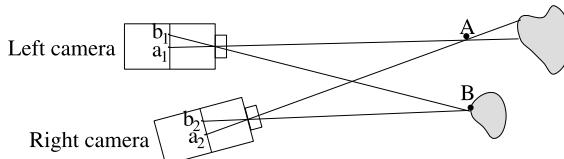


Fig. 1. Pixels b_1 and b_2 , projections of fly B, have identical grey levels, while pixels a_1 and a_2 , projections of fly A, which receive their illumination from two different physical points on the object's surface, get different grey levels

The fitness function exploits this property and evaluates the degree of similarity of the pixel neighbourhoods of the projections of the fly, giving higher fitness values to those probably lying on objects surfaces:

$$\text{fitness}(\text{indiv}) = \frac{G}{\sum_{\text{colours}} \sum_{(i,j) \in N} (L(x_L + i, y_L + j) - R(x_R + i, y_R + j))^2}$$

- (x_L, y_L) and (x_R, y_R) are the fly's left and right projection coordinates, which can be calculated readily [8] using the camera calibration parameters,
- L and R the corresponding grey level values in the left and right images,
- N is a small (usually 5x5 or 7x7 pixel) neighbourhood.

The numerator G is an image gradient-based normalizing factor designed to reduce the fitness of the flies projecting into uniform regions. The population is initialised randomly inside the intersection of the cameras' fields of view (figure 2). The values of z^{-1} are uniformly distributed between zero (or $1/d_{\max}$) and $1/d_{\min}$.

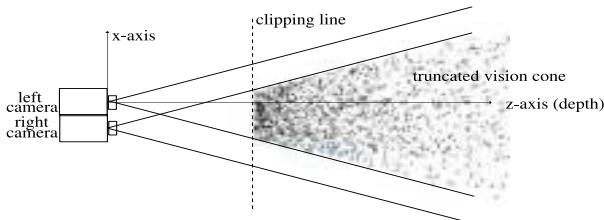


Fig. 2. The fly population is initialised inside the intersection of the cameras fields of view

1.3 The Operators

Several versions of the algorithm have been tested: steady state or generational, using elitist ranking or tournament selection [2], with basically equivalent results. We used the following operators [1]:

2-D sharing reduces the fitness values of flies projecting into crowded areas.

Mutation allows extensive exploration of the search space. It uses an approximation of a Gaussian random noise added to the flies' chromosome parameters (x, y, z).

A *2-parent* and a *3-parent barycentric crossover operator* have been introduced in order to take into account the frequent straight lines or planar surfaces existing in real-world scenes.

An *immigration* operator, similar to the initialisation function, has been introduced as a watchdog to better detect new events in the scene.

1.4 Flies as a Tool for Robot Vision

Compared to other state-of-the-art stereovision algorithms, the Fly algorithm does not provide the most accurate 3-D representation of the environment, to be fair. However, it is extremely fast and robust. As far as we know, no other stereovision algorithm is

able to cope with scene changes while the algorithm is working. Having a progressively improving analysis rather than a blind stage followed by an ultra-accurate 3-D polygonal representation, is highly regarded by the Robotics community. However, most benefits of the fly algorithm would be lost without matching robot controllers.

1.4.1 Processing Image Sequences While Robot Is Moving

We have developed several methods to process stereo image sequences [11]. The simplest one is the *random approach*, which consists in keeping the same population evolving through frame changes. Thus, only the images (and therefore the fitness function inputs) are modified when the fly population evolves. When motion is slow enough, using the results of the last step speeds up convergence significantly compared to using a totally new random population. This may be seen as a fly-based collective memory of space, exploiting the similarity between consecutive scenes. It does not alter the simplicity of the static method and does not require any algorithm change, but it is best suited to situations where the cameras are static while there may be movements in the scene. The following stage consists of updating the flies' positions at each generation, using the odometric information available about the robot's motion in order to give better initial conditions and allow faster convergence of the fly population. This was the basis of the *proprioceptive fusion* described in [2].

In the next stage, introduced in this paper, we consider the possibility of reversing the flow of information: rather than injecting external information about the robot's motion into the fly algorithm, it is possible to do it the other way and exploit the shape of the fly population in order to extract information about the robot's motion.

1.4.2 Obstacle Avoidance and Multi-sensor Fusion

The first controllers developed to match the fly algorithm, were aiming at obstacle avoidance. The fastest and most elegant one [2] uses harmonic functions [4], permanently updated by the fly density, in order to generate and update in real time a smooth obstacle avoidance trajectory towards a given target. Dirichlet boundaries are created (1 for high fly densities, 0 for the target position) and the harmonic function is updated. The steering command of the robot is the gradient of the harmonic function.

We have shown it is possible to use several sensors providing independent information sources on the surrounding scene and the robot's position, such as sonars or wheel position coders, and fuse them through the introduction of corresponding additional terms into the fitness function. This sensor fusion technique keeps the main properties of the fly algorithm: asynchronous processing, no low-level image pre-processing or costly image segmentation, fast reaction to new events in the scene.

2 A Robot Understanding Its Environment: The SLAM Problem

2.1 Classical Approaches to SLAM

On a real-world robot, odometric data given by wheel rotation coders give an estimation of the robot's position. In most applications, wheel rotation is under control of the trajectory planner. The actual robot trajectory does not exactly match the target trajectory due to "trajectory noise" caused by factors such as wheel slipping, tyre wear or

uneven ground surface. SLAM (Simultaneous Localisation And Mapping) [5, 6] allows the robot to use its vision capabilities to extract odometry information, progressively build a map of its environment and localise itself within this map. One of the main problems is how to deal with noise and errors on the positions of objects.

2.2 Flies and the SLAM Problem

2.2.1 Defining a Pseudo-Distance between Fly Populations

Due to the evolutionary (and therefore stochastic) nature of the Fly algorithm, two runs on the same stereo image pair will generally not give the same population of flies. In order to achieve localisation based on fly population 3-D patterns, we have to define some sort of “pseudo-distance” between fly populations.

Given two fly populations A and B , we say that $A \equiv B$ (A “equivalent” to B) if A and B are the results of two different runs of the Fly algorithm on the *same* image pair.

Let F be the set of 3-D positive isometries, I the identical transformation.

The pseudo-distance is not required to fulfil the usual properties of distances. In fact, the ideal property we would welcome from a pseudo-distance δ is:

$$\text{if } A \equiv B, \forall f \in F - \{I\}, \delta(A, B) < \delta(A, f(B))$$

In other terms, given two equivalent populations, any displacement applied to one of them should give a greater distance between them.

Let us consider a distance d (e.g. Euclidean) in the affine 3-D space. The Hausdorff distance:

$$\delta(A, B) = \min_{a \in A} \max_{b \in B} d(a, b)$$

is not usable in our application: if applied to two equivalent populations $A \equiv B$ it is extremely sensitive to any single fly missing in either population [12]. What we actually need is a pseudo-distance giving low values for $\delta(A, A)$ but about equally low values for $\delta(A, B)$ if $A \equiv B$. In other terms, $\delta(A, B)$ will have to be low if for each fly $a_i \in A$ there exists a fly $b_i \in B$ close enough to a_i . In order to preserve as much as possible the analytic regularity of the pseudo-distance, we use the harmonic mean:

$$\text{harmmean}(X) = \frac{N}{\sum \frac{1}{x_i}}$$

which is not too sensitive to high values. Therefore, with

$$A = \{a_i\} i \in [1, n], B = \{b_j\} j \in [1, p]$$

we define the pseudo-distance between populations A and B :

$$\delta(A, B) = \frac{p}{n} \sum_i \frac{1}{\sum_j \frac{1}{\varepsilon + d(a_i, b_j)}}$$

The parameter $\varepsilon > 0$ avoids divisions by zero. Obviously, δ is not a distance. It lacks at least two properties of distances, as $\delta(A, B) \neq \delta(B, A)$ and $\delta(A, A) \neq 0$. The real point is to know if and how it can be used for self-localisation.



Fig. 3. The “corridor” scene (stereo pair)



Fig. 4. Flies resulting of one particular run of the Fly algorithm, projected on the left image (printout contrast has been reduced in order to enhance the visibility of flies)



Fig. 5. Projection of the same flies on the right image

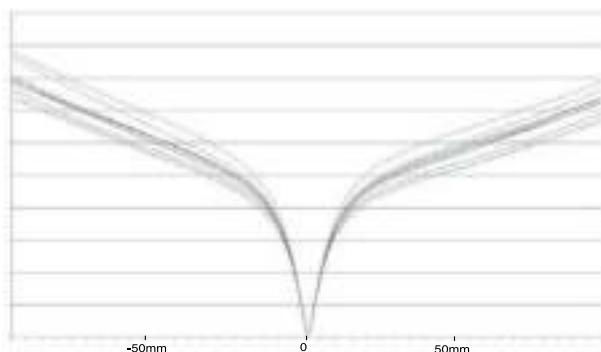


Fig. 6. Pseudo-distances between 10 fly populations and their replicas, in function of translation x . Each graduation on the horizontal axis represents 5 millimetres.

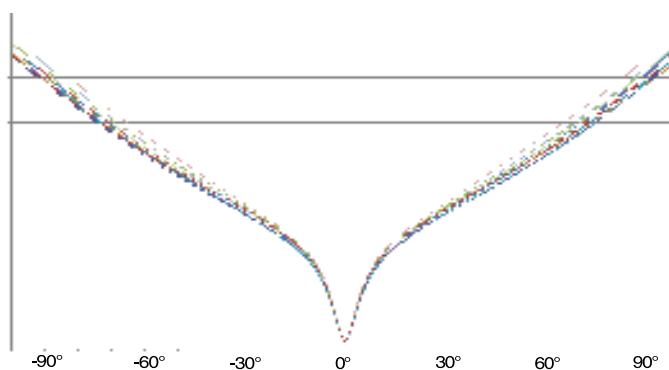


Fig. 7. Pseudo-distances between 10 equivalent populations and their replicas, in function of the rotation angle. Each graduation on the horizontal axis represents about 9 degrees.

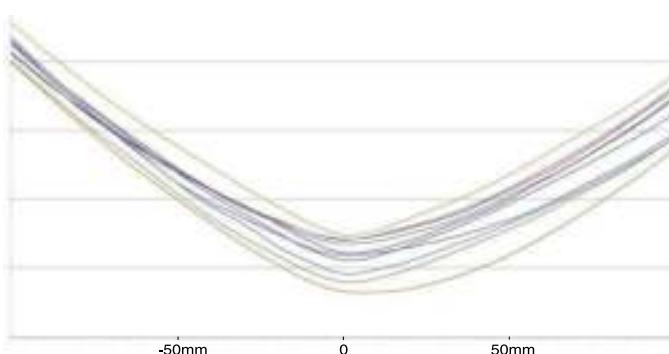


Fig. 8. Pseudo-distances between a reference population and 10 equivalent populations, in function of translation x

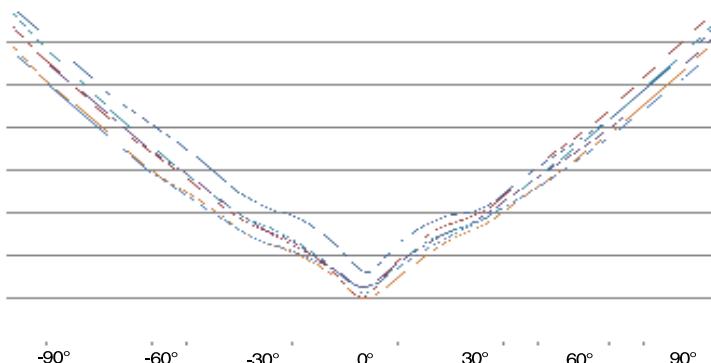


Fig. 9. Pseudo-distances between a reference population and 10 equivalent populations, in function of the rotation angle

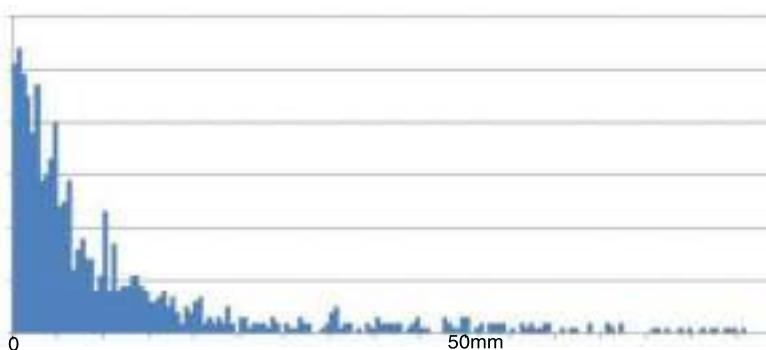


Fig. 10. Histogram of the x - position of the minima of pseudo-distance curves, using 1000 equivalent populations. In this series of experiments, standard deviation is 16 millimetres in x , 15 millimetres in z and 2.77 degrees in θ .

2.2.2 Experimental Properties

We tested this technique with stereo images. First, we studied how the pseudo-distance between one population and the same population behaves when one of the populations is shifted or rotated (figures 6 and 7). Then, we did similar experiments but comparing one reference population to different but equivalent populations (i.e. obtained from the same stereo image pair (figures 8 - 10). In all our experiments, we obtained smooth, convex curves indicating that finding their minimum only requires in practice a basic gradient descent technique. In figure 6, the position of minima shows that fly populations registration based on the detection of pseudo-distance minima will give reasonably small registration errors. A histogram of registration errors, taken on 1000 equivalent fly populations, is given in figure 10, showing a registration error standard deviation of 16 millimetres in x , 15 millimetres in y and 2.77 degrees (0.048 radian) angle.

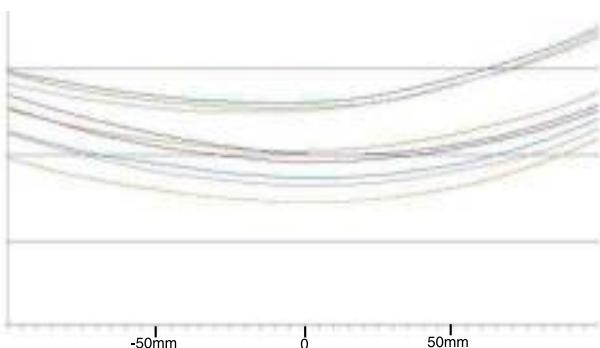


Fig. 11. Pseudo-distances between a population generated from one point of view, and 10 populations generated from another point of view, in function of the translation x from the optimal registration position

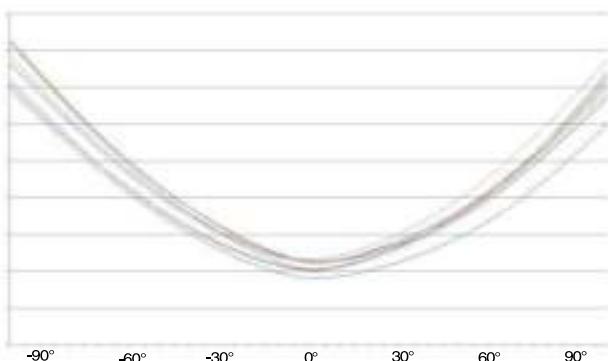


Fig. 12. Pseudo-distances between a population generated from one point of view, and 10 populations generated from another point of view, in function of the rotation angle around the optimal registration position

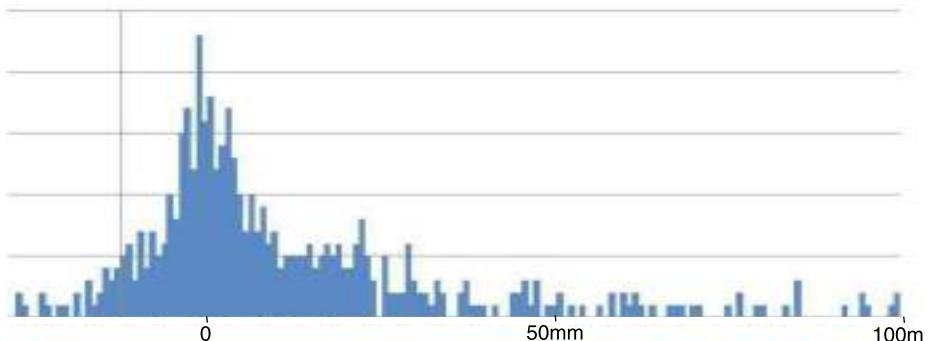


Fig. 13. Histogram of the x -position of the minima of pseudo-distance curves. We took 1000 instances of fly populations generated with the first stereo pair, then for each one we generated an instance of the fly population from the second stereo pair and calculated the position of the optimal (x, z, θ) . The maximum corresponds to the value of the actual translation x . The histograms with z and θ are similar.

3 Fly Populations Registration

The next step is now, rather than correlating equivalent fly populations, to examine how the pseudo-distance can be used to register fly populations produced from stereo pairs taken from different points of view, in order to derive the camera's motion. Here, we are using two stereo pairs from the same scene, taken from different points of view. The camera assembly has been moved horizontally on a trolley, only allowing three degrees of freedom (x , z , θ). Figures 11 to 13 show statistical results of fly population registration. All our experiments on different scene types showed the pseudo-distance remains a convex function of (x, z, θ) which can be minimized using a simple gradient descent on the 3-dimensional space (x, y, θ) . On the example given here, the standard deviation of camera localisation is 19 mm in x , 21 mm in z and 0.85 degree in θ . The diagrams show how this function varies along the coordinate axes.

4 Conclusion

This research on stereovision-based self-localisation using the Fly algorithm opens the way to a new, lightweight mobile robot localisation method using convex optimisation over results obtained by a real-time evolutionary vision algorithm. Further research will be oriented towards a more complete scheme in which the fly populations found during the robot's movement will be stored and used to progressively build a fly-based mapping of the robot's environment. Exploiting a great number of continuously updated fly populations should improve geometrical precision. This will hopefully open a new route to fast resolution of the SLAM problem.

References

1. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimisation, technical report, University of Dortmund (1992)
2. Boumaza, A., Louchet, J.: Using Real-time evolution in Robotics. In: Boumaza, A., Louchet, J. (eds.) EVOIASP 2001, Artificial Evolution in Image Analysis and Signal Processing, Como, Italy (April 2001)
3. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Individual GP: an Alternative Viewpoint for the Resolution of Complex Problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakielka, M., Smith, R.E. (eds.) Genetic and Evolutionary Computation Conference GECCO 1999. Morgan Kaufmann, San Francisco (1999)
4. Connolly, C.I., Burns, J.B., Weiss, R.: Path planning using Laplace's equation. In: Proceedings of the IEEE Conference on Robotics and Automation ICRA 1990, pp. 2102–2106 (May 1990)
5. Davison, A.J.: Real-time Simultaneous Localisation and Mapping with a Single Camera. In: Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 2, pp. 1403–1410 (2003)
6. Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. IEEE Transactions on Robotics and Automation 17(3), 229–241 (2001)
7. Fogel, D.B.: Handbook of Evolutionary Computation. IEEE Press, Los Alamitos (1997)
8. Haralick, R.M.: Using Perspective Transformations in Scene Analysis. Computer Graphics and Image Processing 13, 191–221 (1980)

9. Horn, B.H.: Robot Vision. McGraw-Hill, New York (1986)
10. Louchet, J.: From Hough to Darwin: an Individual Evolutionary Strategy applied to Artificial Vision. In: Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M., Ronald, E. (eds.) AE 1999. LNCS, vol. 1829. Springer, Heidelberg (1999)
11. Louchet, J.: Using an Individual Evolution Strategy for Stereovision. Genetic Programming and Evolvable Machines 2(2), 101–109 (2001)
12. Perez-Garcia, A., Ayala-Ramirez, V., Sanchez-Yanez, R.E., Avina-Cervantes, J.G.: Monte carlo evaluation of the hausdorff distance for shape matching. In: Martínez-Trinidad, J.F., Carrasco Ochoa, J.A., Kittler, J. (eds.) CIARP 2006. LNCS, vol. 4225, pp. 686–695. Springer, Heidelberg (2006)

Genetic Image Network for Image Classification

Shinichi Shirakawa, Shiro Nakayama, and Tomoharu Nagao

Graduate School of Environment and Information Sciences, Yokohama National University, 79-7, Tokiwadai, Hodogaya-ku, Yokohama, Kanagawa, 240-8501, Japan
shirakawa@nlab.sogo1.ynu.ac.jp, shiro@nlab.sogo1.ynu.ac.jp,
nagao@ynu.ac.jp

Abstract. Automatic construction methods for image processing proposed till date approximate adequate image transformation from original images to their target images using a combination of several known image processing filters by evolutionary computation techniques. Genetic Image Network (GIN) is a recent automatic construction method for image processing. The representation of GIN is a network structure. In this paper, we propose a method of automatic construction of image classifiers based on GIN, designated as Genetic Image Network for Image Classification (GIN-IC). The representation of GIN-IC is a feed-forward network structure. GIN-IC transforms original images to easier-to-classify images using image transformation nodes, and selects adequate image features using feature extraction nodes. We apply GIN-IC to test problems involving multi-class categorization of texture images, and show that the use of image transformation nodes is effective for image classification problems.

1 Introduction

Various image processing and recognition techniques using evolutionary computation (EC) have been studied and their effectiveness is demonstrated [1]. The typical examples are template matching, image filter design, image segmentation, and image classification. In particular, genetic programming (GP) [2] has been frequently applied to image classification tasks [3,4,5,6]. Tackett primarily applied GP to solve image classification problems [3]. Parallel Algorithm Discovery and Orchestration (PADO) [4,5], a graph-based GP method, was applied to object recognition problems. Zhang et al. used linear genetic programming (LGP) for multi-class image classification [6]. The results showed that this approach outperforms the basic tree-based GP approach. In addition, GP was used to develop useful texture-feature extraction algorithms [7,8]. In these studies, the evolved features are either at par or outperform human-designed features.

In addition, automatic construction methods for image processing have been proposed [9,10,11,12,13]. They have constructed various types of image processing algorithms automatically. The genetic image network (GIN) is a recent automatic construction method for image processing [12,13]. The representation of GIN is a network structure.

In general, it is difficult to select and extract image features because the appropriate image features depend on the images of a target problem. In this paper, we propose a method for automatic construction of image classifiers based on GIN, the Genetic Image Network for Image Classification (GIN-IC). The representation of GIN-IC is a feed-forward network structure. GIN-IC is composed of image transformation nodes, feature extraction nodes, and arithmetic operation nodes. GIN-IC transforms the input images to easier-to-classify images using the image transformation nodes and selects adequate image features using the feature extraction nodes. We apply GIN-IC to test problems of multi-class classification of texture images.

The next section of this paper is an overview of several related studies. In Section 3, we describe our proposed method, GIN-IC. Next, in Section 4, we apply the proposed method to the problem of texture classification and show several experimental results. Finally, in Section 5, we describe conclusions and future work.

2 Related Work

2.1 Genetic Programming and Graph-Based Genetic Programming

Genetic Programming (GP) [2], an evolutionary computation technique, was introduced by Koza. GP evolves computer programs, which are usually tree structures, and searches for a desired program using a genetic algorithm (GA). Recently, many extensions and improvements to GP have been proposed. Parallel Algorithm Discovery and Orchestration (PADO) [4,5] is a graph-based GP rather than a tree structure. PADO was applied to object recognition problems. Cartesian genetic programming (CGP) [14] was developed from a representation used for the evolution of digital circuits and it represents a program as a graph. In CGP, the genotype is an integer string that denotes a list of node connections and functions. This string is mapped into the phenotype of an index graph.

2.2 Automatic Construction of Image Transformation

In image processing, it is difficult to select filters satisfying the transformation from original images to their target images. The Automatic Construction of Tree structural Image Transformation (ACTIT) system [9,10,11] has been proposed previously. ACTIT approximates adequate image transformation from original images to their target images by a combination of several known image processing filters. ACTIT constructs tree-structured image processing filters using GP [2]. The individual in ACTIT is a tree-structured image transformation. The terminal nodes of a tree are the original images, whereas the non-terminal nodes are several kinds of image processing filters. A root node represents an output image. Users provide training images, and the ACTIT system automatically constructs appropriate image processing procedures. 3D-ACTIT [10,11] is an extended method that automatically constructs various 3D image processing

procedures, and is applied to medical image processing and video image processing. Recently, two other extensions of ACTIT, Genetic Image Network (GIN) [12] and Feed Forward Genetic Image Network (FFGIN) [13], have been proposed. Instead of a tree representation, they are represented by a network structure. Their biggest difference from ACTIT is the structure of connections between image processing filters. In general, a network structure theoretically includes a tree structure (i.e., a network structure also represents a tree structure). Therefore, the descriptive ability of a network representation is higher than that of a tree structure. In GIN and FFGIN, a genotype is a string of integers that indicate image processing filter types and connections. Other studies show that GIN and FFGIN automatically construct a simple structure for complex image transformation using their network representation [12,13].

3 Genetic Image Network for Image Classification (GIN-IC)

3.1 Overview

The image classifier GIN-IC consists of image transformation, feature extraction, and arithmetic operation components. The greatest advantage of GIN-IC is its image transformation (preprocessing) component, which influences image feature selection. In other words, GIN-IC generates and selects adequate image features by a combination of nodes.

3.2 Structure of GIN-IC

GIN-IC constructs an acyclic network-structured image classifier automatically. Figure 1 shows an example of the phenotype (feed-forward network structure) and genotype (string representing the phenotype) of the proposed method. One of the benefits of this type of representation is that it allows the implicit reuse of nodes in its network. The nodes of GIN-IC are categorized into five types: input nodes, image transformation nodes, feature extraction nodes, arithmetic operation nodes, and output nodes. Input nodes correspond to original images. Image transformation nodes execute image transformation using the corresponding well-known image processing filters. We prepare one-input one-output filters and two-inputs one-output filters in the experiments. Feature extraction nodes extract an image feature from input images. Arithmetic operation nodes execute arithmetic operations. Image classification is performed using the values of output nodes. The image classification procedure in GIN-IC is as follows:

1. Image transformation (preprocessing) of original images
2. Feature extraction from images
3. Arithmetic operation and image classification

In GIN-IC, these processes evolve simultaneously.

In GIN-IC, the feed-forward network structure of nodes is evolved, as shown in Figure 1. Numbers are allocated to each node beforehand. Increasingly large

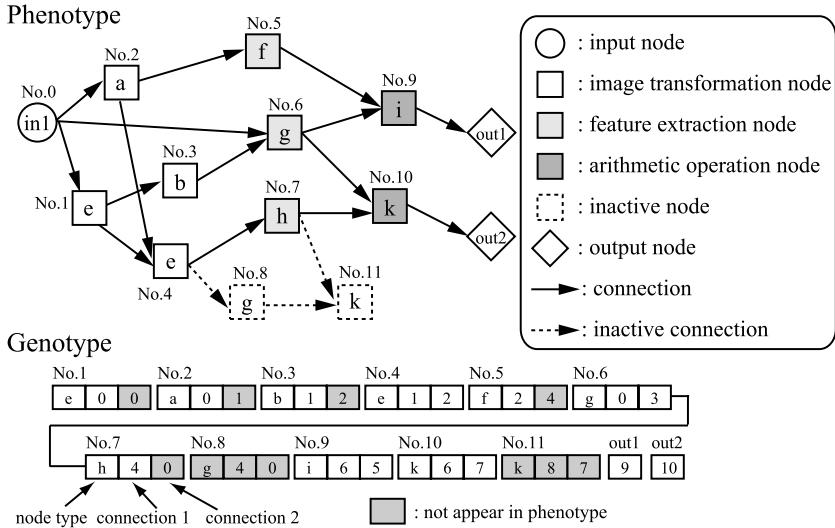


Fig. 1. Example of a structure in GIN-IC (phenotype) and the genotype, which denotes a list of node types and connections

numbers are allocated, in order, to input nodes, image transformation nodes, feature extraction nodes, arithmetic operation nodes, and output nodes. Connections that cannot be executed are restricted at the genotype level; for instance, the feedback structure is restricted. The nodes take their input from the output of previous nodes in a feed-forward manner. Furthermore, the image transformation and feature extraction nodes are connected either with input nodes or image transformation nodes. The arithmetic operation nodes are connected either with the feature extraction nodes or other arithmetic operation nodes; that is, the connections of nodes obey these data type restrictions, which are based on the idea of strongly typed genetic programming [15].

To adopt an evolutionary method, GIN-IC uses genotype-phenotype mapping. This genotype-phenotype mapping method is similar to Cartesian Genetic Programming (CGP). The feed-forward network structure is encoded in the form of a linear string. The genotype in GIN-IC is a fixed length representation and consists of a string that encodes the node function ID and connections of each node in the network. However, the number of nodes in the phenotype can vary in a restricted manner, as not all of the nodes encoded in the genotype have to be connected. This allows the existence of inactive nodes. In Figure 1, node No. 8 and 11 are inactive nodes.

Because GIN-IC constructs a feed-forward network structured image classification procedure, it can represent multiple outputs. Therefore, GIN-IC enables easy construction of a multi-class image classification procedure using a single network structure.

3.3 Genetic Operator and Generation Alteration Model

To obtain the optimum structure, an evolutionary method is adopted. The genotype of the proposed method is a linear string. Therefore, it is able to use a standard genetic operator. In this paper, we use mutation as the genetic operator. The mutation operator affects one individual, as follows:

- Select several genes randomly according to the mutation rate P_m for each gene.
- Change the selected genes randomly under the constraints of the structure.

We use (1+4) Evolution Strategy ((1+4) ES) as the generation alternation model. The (1+4) ES procedure in the experiments works as follows:

1. Set generation counter $t = 0$. Generate an individual randomly as a parent M .
2. Generate a set of four offsprings, C , by applying the mutation operation to M .
3. Select the elite individual from the set $M + C$ (the offspring is selected if the same best fitness with the parent). Then replace M with the elite individuals.
4. Stop if a certain specified condition is satisfied; otherwise, set $t = t + 1$ and go to step 2.

Since GIN-IC has inactive nodes, a neutral effect on fitness is caused by genetic operation (called neutrality [16,14]). In step 3, the offspring is selected if the same best fitness with the parent, then the searching point moves even if the fitness is not improved. Therefore, we believe that efficient search is achieved though simple generation alternation model. This (1+4) ES is also adopted and showed the effectiveness in CGP [14] which has feed-forward network structure as well as GIN-IC.

4 Experiments and Results

In this section, we apply GIN-IC to the problem of multi-class texture classification and confirm that using image transformation nodes is effective.

4.1 Settings of the Experiment

In this experiment, we apply GIN-IC to a simple test problem of multi-class classification of texture images. We use texture images from the publicly available database VisTex¹. We use six classes in this experiment. We make 128 images with 64×64 pixels each by dividing two texture images of 512×512 pixels for each class. All images used in the experiment are grayscale images with a size of 64×64 pixels. The number of training images is 60 (10 images for each class). The training images used in the experiment are displayed in Figure 2.

¹ <http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html>

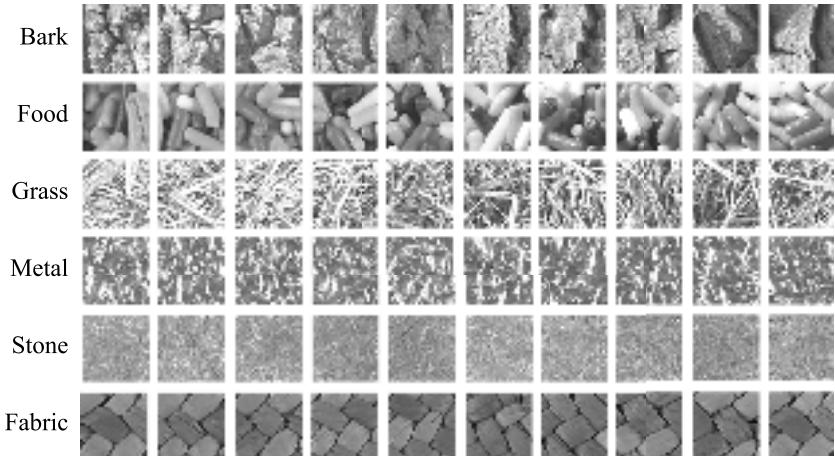


Fig. 2. Training images used in the experiment. There are six classes: Bark, Food, Grass, Metal, Stone, and Fabric.

Table 1. Parameters used in the experiment

Parameter	Value
Generation alternation model	(1+4)ES
Number of generations	112500
Mutation rate (P_m)	0.02
Number of image transformation nodes	100 or 0
Number of feature extraction nodes	100
Number of arithmetic operation nodes	100
Number of output nodes	6

To take advantage of the multi-class image classification capabilities of GIN-IC, we prepared six output nodes, one for each class. The class represented by the output node with the largest value is considered to be the class of the input image.

We use the number of correct classifications of training images and the number of active nodes as a fitness function. The fitness function used in these experiments is described as follows:

$$f = N_c + \frac{1}{N_a} \quad (1)$$

where N_c is the number of correct classifications of training images, and N_a is the number of active nodes. A higher numerical value indicates better performance. If the classification performance is the same, the number of active nodes should be small in this fitness function. We think N_c is more important than N_a . However, a weight coefficient may have to be given to N_c and N_a .

The parameters used by the proposed method are shown in Table 1. We assume that the number of image transformation nodes is either 0 (to verify the effect of using image transformation nodes) or 100. If the number of the image transformation nodes is 0, this method resembles classification techniques using the usual GPs. Results are given for ten different runs with the same parameter set.

We prepare simple and well-known image processing filters as the image transformation nodes in the experiment (26 one-input one-output filters and 9 two-input one-output filters), e.g., mean filter, maximum filter, minimum filter, Sobel filter, Laplacian filter, gamma correction filter, binarization, linear transformation, difference, logical sum, logical prod, and so on. Seventeen simple statistical values are used as feature extraction nodes, e.g., mean value, standard deviation, maximum value, minimum value, mode, 3 sigma in rate, 3 sigma out rate, skewness, kurtosis, and so on. The arithmetic operation nodes are 20 well-known arithmetic operations, e.g., addition, subtraction, multiplication, division, threshold function, piecewise linear function, sigmoid function, absolute value, equalities, inequalities, constant value, and so on. GIN-IC is expected to construct a complex image classifier using a combination of these nodes.

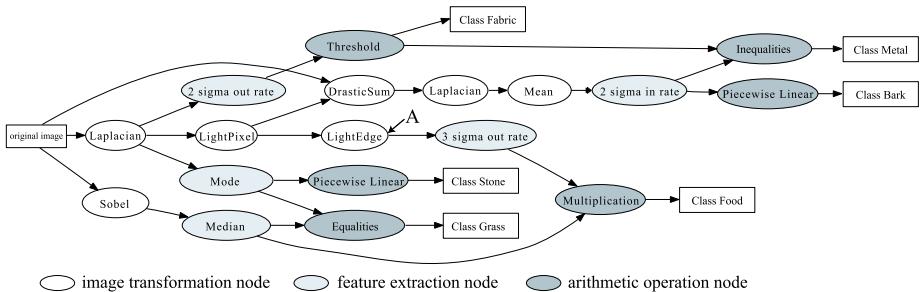
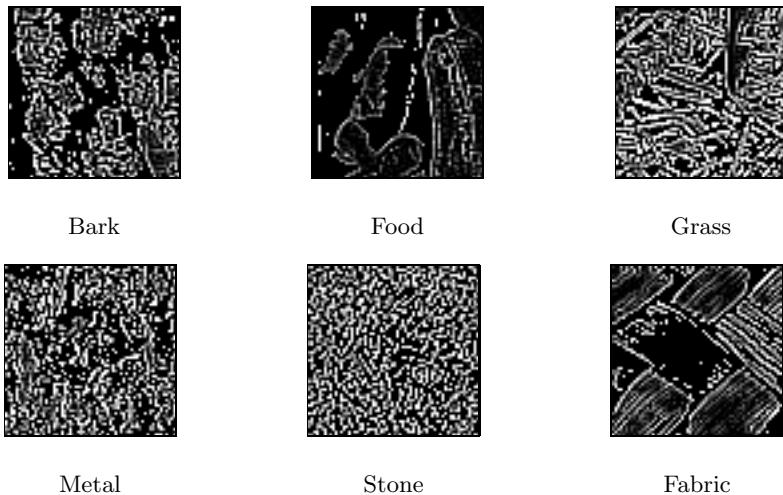
4.2 Results and Discussion

When the number of image transformation nodes was 100, GIN-IC achieved 100% classification accuracy for all training images. However, 100% accuracy in classifying training images was not obtained when the number of image transformation nodes is 0, suggesting the effectiveness of using image transformation nodes. We believe that the image transformation nodes change the original images to images adequate for classification.

Next, we apply the obtained classifiers to test images. Test images (non-training images that are similar to training images) are not used in the evolutionary process. The number of test images is 708 (118 images for each class). The average classification performance over ten different runs for the test images are shown in Table 2. The values of Table 2 are discrimination rates for each class. According to the results, the obtained classifiers affect test images as well

Table 2. Discrimination rate for the test images using obtained classifiers (average over 10 different runs)

	Use image filters	Do not use image filters
Bark	70.7 %	53.4%
Food	86.1 %	44.1%
Grass	88.1 %	84.8%
Metal	76.7 %	92.5%
Stone	80.4 %	83.4%
Fabric	94.7 %	92.0%
Average	82.8 %	75.0%

**Fig. 3.** Example of obtained structure using GIN-IC**Fig. 4.** Examples of output images at node A in Figure 3

as training images. The result also shows the effectiveness of using image transformation nodes. In particular, the discrimination rates of “Bark” and “Food” improve when image transformation nodes are used. The discrimination rates of “Metal” and “Stone” are worse on the other hand.

Figure 3 is an example of the obtained structure (feed-forward network-structured image classifiers) constructed by GIN-IC. GIN-IC constructs the structure using the image transformation nodes in its network. Examples of output images at node A are shown in Figure 4. From these output images, GIN-IC transforms the original images to other types of images and also transforms the features of the images. This shows that GIN-IC automatically constructs image transformation and feature extraction components through learning. Moreover, node A connects to the output node of “Food” and the output image of “Food” is characteristic of other class images.

Table 3 shows the confusion matrix and discrimination rates using the structure of Figure 3 for the test images. This classifier achieves a discrimination rate

Table 3. Confusion matrix and discrimination rates using the structure of Figure 3 for the test images. Each column of the matrix represents a predicted class, while each row represents an actual class. Each class has 118 test images.

	Bark	Food	Grass	Metal	Stone	Fabric	Discrimination rate
Bark	94	0	0	13	11	0	79.7%
Food	1	116	0	0	0	1	98.3%
Grass	8	0	97	12	1	0	82.2%
Metal	0	0	0	110	8	0	93.2%
Stone	0	0	0	1	117	0	99.2%
Fabric	0	0	0	0	13	105	90.0%
Average discrimination rate							90.4%

greater than 90%. We can observe that the classifier tends to mistake the “Bark” class for either the “Metal” or “Stone” class.

In the experiments, we only use simple statistical values as image features. We consider that the classification performance improves if the appropriate texture features are prepared.

5 Conclusions and Future Work

In this paper, we propose a new method for automatic construction of an image classifier that evolves feed-forward network structured image classification programs. The image classifier of the proposed method consists of image transformation, feature extraction, and arithmetic operation components. The greatest advantage of the proposed method is the presence of the image transformation (preprocessing) component. We applied the proposed method to the problem of texture classification and confirmed that it obtained the optimum solution. From the experimental results, the obtained classifier is effective in texture classification. We are, however, recognizing that GIN-IC should be compared with other classifiers in order to evaluate the quality of GIN-IC. In future work, we will apply the proposed method to other problems of image classification and object recognition, particularly to larger problems and other types of problems. Moreover, we will introduce the use of images’ color information for classification.

References

1. Cagnoni, S., Lutton, E., Olague, G. (eds.): Genetic and Evolutionary Computation for Image Processing and Analysis. EURASIP Book Series on Signal Processing and Communications, vol. 8. Hindawi Publishing Corporation (2007)
2. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
3. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In: Proceedings of the 5th International Conference on Genetic Algorithms (ICGA 1993), pp. 303–309. Morgan Kaufmann, San Francisco (1993)

4. Teller, A., Veloso, M.: Algorithm evolution for face recognition: What makes a picture difficult. In: International Conference on Evolutionary Computation, Perth, Australia, pp. 608–613. IEEE Press, Los Alamitos (1995)
5. Teller, A., Veloso, M.: PADO: A new learning architecture for object recognition. In: Ikeuchi, K., Veloso, M. (eds.) *Symbolic Visual Learning*, pp. 81–116. Oxford University Press, Oxford (1996)
6. Zhang, M., Fogelberg, C.G.: Genetic programming for image recognition: An LGP approach. In: Giacobini, M. (ed.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 340–350. Springer, Heidelberg (2007)
7. Lam, B., Ciesielski, V.: Discovery of human-competitive image texture feature extraction programs using genetic programming. In: Deb, K., et al. (eds.) *GECCO 2004*. LNCS, vol. 3103, pp. 1114–1125. Springer, Heidelberg (2004)
8. Aurnhammer, M.: Evolving texture features by genetic programming. In: Giacobini, M. (ed.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 351–358. Springer, Heidelberg (2007)
9. Aoki, S., Nagao, T.: Automatic construction of tree-structural image transformation using genetic programming. In: Proceedings of the 1999 International Conference on Image Processing (ICIP 1999), Kobe, Japan, vol. 1, pp. 529–533. IEEE, Los Alamitos (1999)
10. Nakano, Y., Nagao, T.: 3D medical image processing using 3D-ACTIT; automatic construction of tree-structural image transformation. In: Proceedings of the International Workshop on Advanced Image Technology (IWAIT 2004), Singapore, pp. 529–533 (2004)
11. Nakano, Y., Nagao, T.: Automatic construction of moving object segmentation from video images using 3D-ACTIT. In: Proceedings of The 2007 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2007), Montreal, Canada, pp. 1153–1158 (2007)
12. Shirakawa, S., Nagao, T.: Genetic image network (GIN): Automatically construction of image processing algorithm. In: Proceedings of the International Workshop on Advanced Image Technology (IWAIT 2007), Bangkok, Thailand (2007)
13. Shirakawa, S., Nagao, T.: Feed forward genetic image network: Toward efficient automatic construction of image processing algorithm. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Paragios, N., Tanveer, S.-M., Ju, T., Liu, Z., Coquillart, S., Cruz-Neira, C., Müller, T., Malzbender, T. (eds.) *ISVC 2007, Part II*. LNCS, vol. 4842, pp. 287–297. Springer, Heidelberg (2007)
14. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) *EuroGP 2000*. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
15. Montana, D.J.: Strongly typed genetic programming. *Evolutionary Computation* 3(2), 199–230 (1995)
16. Koza, J.R.: *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge (1994)

Multiple Network CGP for the Classification of Mammograms

Katharina Völk, Julian F. Miller, and Stephen L. Smith

The University of York, Department of Electronics, Heslington, York YO10 5DD, UK
`{kv500, jfm7, sls5}@ohm.york.ac.uk`

Abstract. This paper presents a novel representation of Cartesian genetic programming (CGP) in which multiple networks are used in the classification of high resolution X-rays of the breast, known as mammograms. CGP networks are used in a number of different recombination strategies and results are presented for mammograms taken from the Lawrence Livermore National Laboratory database.

Keywords: Evolutionary algorithms, Cartesian genetic programming, mammography.

1 Introduction

Breast cancer is one of the leading causes of death in women in the western world. In 2005, some 40,000 new cases were detected in the UK alone and 10,300 women died in 2005 as a result of the disease [1] making breast cancer the most common cancer in women and the second most common cause of cancer death. The number of breast cancer related deaths has fallen since a screening programme was introduced in 1988. Nonetheless, it is predicted that one in seven woman will develop breast cancer at some point during their life [2].

The detection of breast cancer in the early stages of the disease significantly increases the survival rate of patients. The main method for screening patients is the mammogram, a high resolution x-ray of the breast. The process of identifying and evaluating signs of cancer from mammograms is a very difficult and time-consuming task that requires skilled and experienced radiologists. This assessment is also, by its nature, highly subjective and susceptible to error, leading to cancers being missed and the patients misdiagnosed. To achieve a more accurate and reliable diagnosis, Computer Aided Detection (CAD) systems have been investigated which provide an objective, quantitative evaluation. CAD systems have the potential to help in two main ways: (i) the detection of suspicious areas in the mammogram that require further investigation and (ii) the classification of such areas as cancerous (malignant) or non-cancerous (benign).

The aim of the work reported in this paper is to assess the potential benefit of a new representation of using evolutionary algorithm in the classification of mammograms as part of a CAD system and determine whether further development of such algorithms will lead to a more confident diagnosis. The implementation of a

full CAD system is a huge undertaking and not viable or necessary for the evaluation of the algorithms proposed. Therefore, rather than develop a complete CAD system that acquires, preprocesses and segments appropriate sections of the mammogram, this investigation will rely on prior knowledge by using previously acquired and processed images of known pathology. Thus, only small sub-images taken from previously diagnosed mammograms are used where the nature and location of the suspicious regions are known and have been documented as such by clinical personnel.

The problem presented to our algorithm reduces to one of deciding if the suspicious area is an indication of cancer (malignant) or harmless (benign). Two powerful indicators of cancer that are commonly used in evaluating mammograms are known as masses and microcalcifications. Masses are the larger of the two indicators and can be either benign or malignant. Characteristics such as the border and density of the mass, which is greater for malignant examples, can be used for classification. Traditionally, masses are more difficult to classify than microcalcifications. Microcalcifications are essentially small calcium deposits which occur as the result of secretions from ductal structures that have thickened and dried. They can have a great variety of mostly benign causes, but might also be an indication for malignancy. They are fairly common on mammograms and their appearance increases with age, so that they can be found in 8% of mammograms of women in their late 20s and in 86% of mammograms of woman in their late 70s [2]. Microcalcifications that indicate malignancy are usually less than 0.5mm in size and often grouped into clusters of five or more. Any calcification larger than 1mm is almost always benign [2].

Features that have previously been used to distinguish benign and malignant microcalcifications include their shape, density, distribution and definition. Not only are these characteristics useful for a radiologist attempting to classify a mammogram, but they have been used extensively in feature extraction for established image processing techniques.

Although work by the authors indicate that evolutionary algorithms can be used effectively to analyze masses it was decided, initially, to work exclusively with microcalcifications, as more work has already been done in this area, providing a greater source of literature to which comparisons can be made. Additionally, microcalcifications are easier to identify than masses and so provide a more reliable source of data for both training and testing the algorithms.

Due to the nature of the mammograms, traditionally there is a need for pre-processing, particularly for detection or classification where methods such as wavelets or morphological filtering are used. One of the advantages of using an evolutionary method such as Cartesian Genetic Programming (CGP) is that there is no need for prepossessing the images.

Previous work undertaken in the classification of microcalcifications using evolutionary algorithms is considered in Section 2. The evolutionary algorithm used in the current work will then be described in Section 3 and results applying this technique to a number of digitized mammograms will be considered in Section 4. Finally, the potential of the proposed algorithm will be evaluated in Section 5.

2 Previous Work

Over recent years there has been much research into the application of computer aided diagnosis to breast cancer with numerous different approaches being exploited. Many of these involve image analysis of the digitized mammogram – a low dose x-ray of the breast. A typical approach is to use a pattern recognition scheme that employs (i) sensing, (ii) segmentation, (ii) feature extraction, (iv) feature selection and (v) classification, to isolate and then characterize a feature of interest [3]. Each stage of this processing is a potentially complex operation requiring much investigation.

The work presented in this paper is concerned specifically with the characterization and classification of microcalcifications - the feature extraction, feature selection and classification stages of the pattern recognition scheme [4][5]. Consequently, the sensing and segmentation stages of the scheme, while relevant and important in a fully implemented system [6][7], are not considered here and for the purpose of the experiments described in Section 4 will be undertaken manually.

Evolutionary algorithms are a family of population based algorithms that use facets of biological evolution such as natural selection, reproduction, mutation and recombination to evolve solutions to problems. Examples of evolutionary algorithms include Genetic Algorithms, Genetic Programs are considered below.

Genetic algorithms (GAs) have previously been used in CAD schemes and they have proved successful. One of the key papers that influenced this project is a GA based paper [8] in which a genetic algorithm was used for feature selection and it proved successful in this area. Performance was found to be a match for the well established LDA method and even better sometimes. The review paper [9] also reported the only use of genetic algorithms as being in feature selection as in the aforementioned paper. Neural networks are another biologically inspired technique that has been widely adopted and successfully, but uses of GAs are limited and this raises the question of whether genetic algorithms could be further used. Genetic Programs (GPs) have previously been used in image processing by Cai, Smith and Tyrrell for noise removal from images [10]. In this case a form of genetic program called Cartesian Genetic programming (CGP) was used (this will be explained shortly). Clearly, the removal of noise is a very different to pattern recognition but it suggests that application of genetic algorithms to this type of problem could be an interesting avenue to explore.

An example of the use of genetic algorithms as an alternative feature selection method starts with a data structure termed a chromosome which is the length of the total number of features available. Each gene in the chromosome is a bit which is 1 or 0 where 1 indicates that a particular feature is included. For example bit 5 might be chosen to represent image entropy. There is a population of random chromosomes and for each one classification is performed. A new population is generated using: parent selection, crossover and mutation. When the parents are selected it is designed so that ones deemed fitter are more likely to be chosen. By fitter it is meant the ones that resulted in a more accurate classification. This is continued for either a certain number of population generations or until a certain level of classification is obtained. It should be noted that there might be bias in the classifier, such that a certain set of input values might favor a particular set of features; to avoid this, the broadest range of data sets should be used.

3 Implementation

Cartesian Genetic Programming (CGP) has been used in this work [11]. CGP is a graph-based genetic programming system which has been shown to perform well within a wide range of problem domains. A CGP solution consists of an n -dimensional grid (where n is typically 1 or 2) in which each grid location contains a function. Program inputs and outputs are delivered to and taken from specific grid cells. Interconnections between functions, inputs and outputs are expressed in terms of the grid's Cartesian co-ordinate system. The 2-D CGP general form is shown in Figure 1.

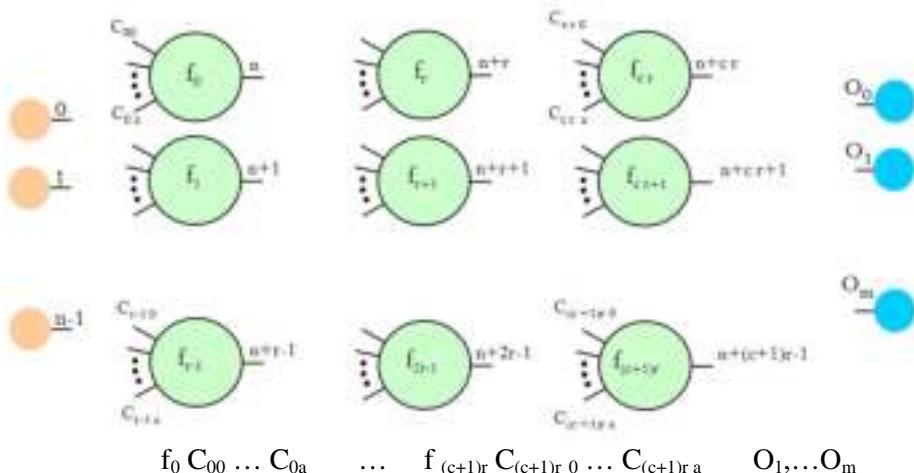


Fig. 1. General form of two-dimensional CGP. It is a grid of nodes whose functions are chosen from a set of primitive functions. Each node is assumed to take as many inputs as the maximum function arity a . Every data input and node output are labelled consecutively (starting at 0) which gives it a unique data address which specifies where the input data or node output value can be accessed (shown in the figure on outputs of inputs and nodes). Nodes in columns cannot be connected to each other. In most cases the graph is directed (as in this paper) so that a node may only have its inputs connected to either input data or the output of a node in a previous column. In general there may be a number of output genes (O_i) which specify where the program outputs are taken from. The structure of the genotype is seen below the schematic. All node functions genes f_i are integer addresses in a look-up table of functions. All connection genes C_{ij} are integers taking values between 0 and the address of the node at the bottom of the previous column of nodes.

A mutation operator can alter both the function present within a grid cell and the connections between components. The efficacy of CGP has been attributed to both implicit reuse of sub-expressions (due to its graphical representation) and its use of functional redundancy [11][14]. In this paper multiple CGP networks are used to evolve a single mammogram. The image is divided into equal sized, non-overlapping parts and each one of these is assigned its own CGP network which is evaluated independently as shown. The division of the mammogram into parts should not, however, be confused with a conventional image processing windowing operation.

More specifically, in this work each genotype consists of 256 independent CGP chromosomes using a grid of 32 rows and 128 columns. Each chromosome has 64 inputs corresponding to an 8x8 grid of grey scale pixel values (0 to 255). Each chromosome has a single output gene. The mammogram images are divided into 256, 8x8 pixel areas. The 64 grey scale pixel values (0 to 255) for each of these areas form the inputs to an individual CGP network, encoded by a chromosome. A summary of these details is given in Table 1. The chromosome mutation rate defines the percentage of genes in each chromosome that are mutated when a genotype is mutated (i.e. for 32x128 nodes there are $3 \times 32 \times 128 + 1$ genes. The one corresponds to the single output gene). The function set is shown in Table 1 (where $x \& y$ represents the bitwise AND function). The output from all node operations is kept within the range 0 to 255, by truncation.

Table 1. Parameters for multiple CGP network

Parameter	Value
No. parts per image	256 (16x16)
Part size	8x8 pixels
Chromosome rearrangement rate	3%
Chromosome mutation rate	1%
No. runs	10
No. generations	1000
No. columns in each CGP network	128
No. rows in each CGP network	32
Function set	$x, x+y, \text{abs}(x-y), \text{abs}(2x-y), x \& y, \text{largest}(x,y), \text{smallest}(x,y)$

One of the motivations for developing this representation was to implement recombination in CGP but at a whole chromosome level as opposed to recombining genes within chromosomes. This multi-chromosome approach has been shown to have a number of advantages [12]. So whole CGP chromosomes could be exchanged rather than components (genes) within one network. This approach not only aims to improve the system's performance on one given part of a mammogram but also allows for improvement of the system's performance on the whole mammogram by allowing individual CGP chromosomes to be swapped and reused for other parts of the image depending on the success or fitness they achieve.

The evolutionary algorithm is a 1+2-ES, in which there are three genotypes (each consisting of 256 chromosomes). There will be one parent selected whose genotype will consist of 256 chromosomes, each of which is the best chromosome chosen from the three population members. However, we have investigated another mutational step after this best genotype is assembled. According to another mutation rate, which we call a re-arrangement mutation rate (see table 1), chromosomes may undergo either a swap or replacement with another of the 256 chromosomes, chosen at random. Specifically:

- (i) a random swap in which any chromosome might be swapped with another (random-swap);
- (ii) a neighbouring swap in which a part might only be swapped at random with its four direct spatial neighbours. The neighbouring swap has been implemented to target structures that continue from one part of the image

- to the next. Neighbouring parts might therefore have similar image properties and are therefore likely to respond equally well to the same chromosome. (neighbourhood swap);
- (iii) a copying operation where a random chromosome is chosen to overwrite a different chromosome (re-use).

If after a rearrangement a chromosomes fitness declines from the operations then the rearrangement is disallowed. If, however, the rearranged part makes an improvement to the resulting fitness, then the exchange is preserved. Although there is a risk that the diversity of chromosomes might be reduced by deleting ones that do not perform well and substituting them with a copies of a fitter one, this approach gives the genotypes a higher opportunity for individual mutation which in itself has the potential of restoring diversity to some extent. We can see this because if every chromosome is unique (no copies) then mutations can only be beneficial independently. If there are duplicated chromosomes, any mutation occurring in those would have to be, on average, beneficial to all of them. This means one chromosomes fitness might be reduced if all other copies gained a higher fitness, through the rearrangement.

Images used in this study are constructed from mammograms in the LLNL database that feature microcalcifications. As described in the introduction the images have been manually edited to avoid the need to automatically locate microcalcifications. In each case a 128x128 pixel image is constructed containing at least one microcalcification from a particular mammogram. Each image is then logically divided into 256 parts and the status of each part labelled as either being benign or malignant according to the radiologist.

When an image is processed by the system the output value generated for each chromosome by its respective CGP network is compared to a predetermined threshold. An output value above the threshold is interpreted as an indication of benignity and an output value below the threshold an indication of malignancy. In this study output values ranged from 0 to 255 and the threshold adopted was 4. This bias toward benign results reflects the relative scarcity malignant areas within the image. A fitness value can be then be calculated on this basis of this predicted value and the predetermined status of that part of the image as identified by the radiologist.

4 Results

As previously stated, the mammograms used in this study were taken from the Lawrence Livermore National Laboratory database [13] that specifically featured microcalcifications. The mammograms were cropped to images of 128x128 pixels containing at least one microcalcification. In total 31 images were created, of which 13 contained malignant microcalcifications and 18 benign microcalcifications. Some 67% of these images were used for training the CGP network and the remaining 33% for the testing stage.

In the training phase, the single parts reached different fitness values with an average of 81.2% to 90.6% depending on the method used. Full results are given in Table 2 which also details the performance of a number of chromosome rearrangement strategies. Graphs for average and best fitness are also given in Figures 2 and 3 respectively.

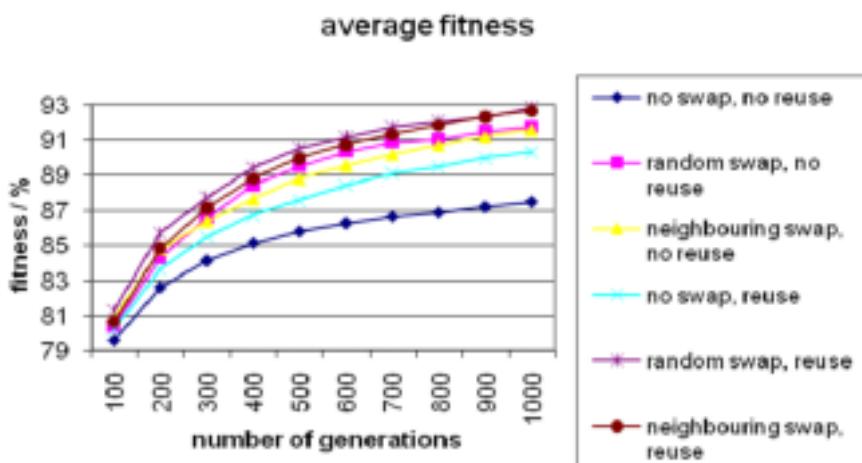
Table 2. Fitness values for training of CGP networks

Recombination	Best part's fitness (%)	Worst part's fitness (%)	Average fitness (%)
No swap, no reuse	94.9	54.5	81.2
No swap, reuse	95.7	67.8	85.5
Neighbouring swap, no reuse	96.9	62.0	87.8
Neighbouring swap, reuse	96.9	63.9	89.0
Random swap, no reuse	96.5	70.6	89.4
Random swap, reuse	96.9	71.4	90.6

One of the problems that might occur when applying the evolved programme to test images is that some of the CGP networks may not have been trained with image parts containing a microcalcification and therefore will only recognise background breast tissue. To overcome this problem each part of the test image is evaluated with every evolved CGP chromosome. The highest fitness score generated is then used to classify that respective part (without any knowledge of its true class). The fitness values of the test set of images are shown in Table 3.

Table 3. Fitness values for test images

	Best part's fitness (%)	Worst part's fitness (%)	Average fitness (%)
No swap, no reuse	97.3	84.7	93.3
No swap, reuse	98.0	81.6	94.1
Neighbouring swap, no reuse	98.4	82.0	94.5
Neighbouring swap, reuse	98.4	81.2	94.5
Random swap, no reuse	98.0	82.7	94.5
Random swap, reuse	98.4	81.2	94.9

**Fig. 2.** Average fitness for training phase of CGP networks with different recombination strategies

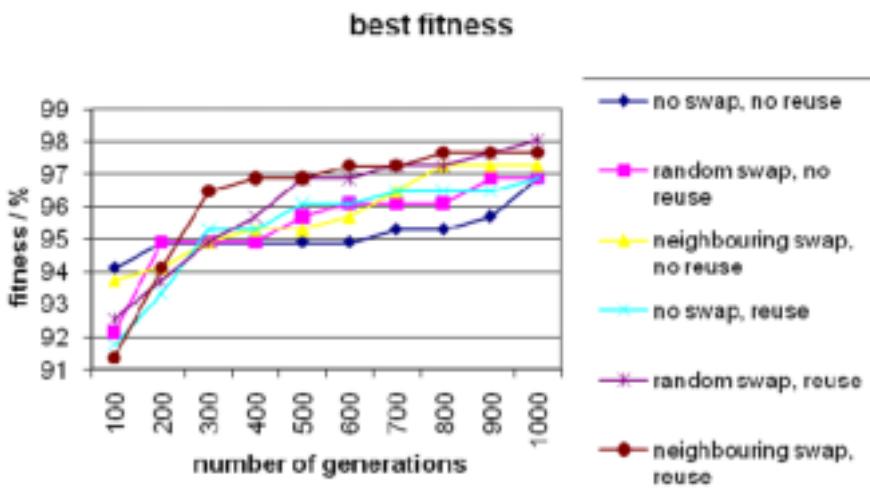


Fig. 3. Best fitness for training phase of CGP networks with different recombination strategies

An example result is shown in Figure 4. Each figure represents an image part (64 pixels), the number “1” indicates a malignancy in that area and number “0” indicates a benignity. The radiologist’s classification of malignancy is indicated by grey shading. Figure 4 shows the respective mammogram.

Fig. 4. System's classification of image parts 1 = malignancy, 0 = benignity. Grey shading radiologist's classification of malignancy.

Fig. 5. Mammogram relating to results shown in Figs. 2 and 3

5 Conclusion

This paper has described a novel multiple network CGP evolutionary algorithm applied to the classification of mammograms. The results presented have demonstrated that the method correctly classifies microcalcifications as being malignant or benign. Given the limitations of the training and the test sets available, and no pre-processing has needed to been applied, the results are very encouraging. The main limitation of the data available is the low number of usable images from a fairly old database. To overcome this problem, new databases of mammograms are being sought so further work to evaluate and improve the system can be conducted.

References

1. Office for National Statistics,
<http://www.statistics.gov.uk/cci/nugget.asp?id=575>
2. Kopans, D.B.: Lippincott Williams & Wilkins (2006)
3. Gonzalez, R., Woods, R.: Digital Image Processing. Prentice-Hall, Englewood Cliffs (2002)
4. Kim, J., Park, H.: Statistical Textural Features for Detection of Microcalcifications in Digitized Mammograms. *IEEE Transactions Medical Imaging* 18(3), 231–238 (1999)
5. Fu, J.C., Lee, S.K., Wong, S.T.C., Yeh, J.Y., Wang, A.H., Wu, H.K.: Image segmentation feature selection and pattern classification for mammographic microcalcifications. *Computerized Medical Imaging and Graphics* 29, 419–429 (2005)
6. Gavrielides, M., Lo, J., Vargas-Voracek, R., Floyd, C.: Segmentation of suspicious clustered microcalcifications in mammograms. *Medical Physics* 27 (2000)
7. Andolina, V., Lille, S., Willison, K.: Mammographic Imaging: A Practical Guide. Lippincott-Raven (1993)
8. Chan, H., Sahiner, B., Lam, K., Petrick, N., Helvie, M., Goodsitt, M., Adler, D.: Computerized analysis of mammographic microcalcifications in morphological and texture feature space. *Medical Physics* 25(10) (1998)
9. Cheng, H.D., Cai, X., Chen, X., Hu, L., Lou, X.: Computer-aided detection and classification of microcalcifications in mammograms: a survey. *Pattern Recognition* 36, 2967–2991 (2003)
10. Cai, X., Smith, S.L., Tyrrell, A.M.: Benefits of Employing an Implicit Context Representation on Hardware Geometry of CGP. In: Moreno, J.M., Madrenas, J., Cosp, J. (eds.) ICES 2005. LNCS, vol. 3637, pp. 143–154. Springer, Heidelberg (2005)
11. Miller, J.F., Thomson, P.: Cartesian Genetic Programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
12. Walker, J., Walker, J.A., Miller, J.F., Cavill, R.: A Multi-chromosome Approach to Standard and Embedded Cartesian Genetic Programming. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 903–910. ACM Press, New York (2006)
13. Lawrence Livermore National Laboratory database, <https://www.llnl.gov/>
14. Miller, J.F., Smith, S.L.: Redundancy and Computational Efficiency in Cartesian Genetic Programming. *IEEE Transactions on Evolutionary Computation* 10, 167–174 (2006)

Evolving Local Descriptor Operators through Genetic Programming

Cynthia B. Perez and Gustavo Olague

Centro de Investigación Científica y de Educación Superior de Ensenada
Ensenada, B.C. 22860 México
`{cbperez,olague}@cicese.mx`

Abstract. This paper presents a new methodology based on Genetic Programming that aims to create novel mathematical expressions that could improve local descriptors algorithms. We introduce the RDGP-ILLUM descriptor operator that was learned with two image pairs considering rotation, scale and illumination changes during the training stage. Such descriptor operator has a similar performance to our previous RDGP descriptor proposed in [1], while outperforming the RDGP descriptor in object recognition application. A set of experimental results have been used to test our evolved descriptor against three state-of-the-art local descriptors. We conclude that genetic programming is able to synthesize image operators that outperform significantly previous human-made designs.

Keywords: Local descriptors, genetic programming, object recognition.

1 Introduction

Nowadays interest point detectors and descriptors have become a popular choice for object detection and recognition. The appealing idea is to detect a high number of local features that provide simplicity, reliability and efficiency for high level tasks. Today object detection and recognition using the paradigm of local features could be summarized with the following three stages: a) Feature extraction through local features, b) Image description of local patches centered around interest points, and c) Image matching using the local information previously computed. This paper focuses on the second stage, which in our particular case is based on the SIFT (Scale Invariant Feature Transform) algorithm, because it has been shown to be the most robust local invariant feature descriptor [2]. Thus, SIFT [3] consists on detecting local keypoints from difference-of-Gaussian (DoG) images, while the local image description is based on the gradient of the image patch. This patch is centered on the keypoint location, rotated with respect to the dominant orientation and scaled to the appropriate size. We propose in this work to name the gradient magnitude that is used in SIFT and its variants as the local image descriptor operator. Thus, we present a genetic programming based approach to synthesize image processing operators, which are used to improve

the SIFT descriptor. Therefore, we include illumination information in the training stage in order to synthesize the mathematical expressions with the aim to outperform our previous RDGP descriptor and human-made descriptors. In this sense, this research follows the work of evolving interest point detectors [4,5] and attempts to solve a more challenging task, that of improving local descriptors. In the past, several researchers have attempted to improve the SIFT descriptor with some success, see [6,7,8,9]. In fact, there is only one work where statistical learning has been applied to study the selection of some SIFT parameters [10], again with some improvements. In order to understand our approach we will describe the experimental testbed that has been created within the computer vision community. Moreover, we tested our RDGP-ILLUM descriptor recognizing an object in an indoor scenario.

The remainder of this paper proceeds as follows: Section 2 gives an explanation of the metric used for optimizing descriptor operators. Section 3 presents a brief overview of our GP learning system. Finally, experimental results are presented in Section 4 and concluding remarks are given in Section 5.

2 Evaluating Descriptors' Operators through the F-Measure

Local descriptors are commonly evaluated in ROC (Receiver Operating Characteristics) or Precision-Recall spaces which are based on metrics that are computed from the contingency table; considering the computed matches between two local regions of two different images. This research focuses on the precision-recall space since ROC space is more appropriate for measuring the performance of object classification approaches, see [11]. Mikolajczyk and Schmid [2] present a state-of-the-art evaluation testbed of local descriptors that is based on recall vs 1-precision space, which is obtained with knowledge of an homography that relates each image pair of the database. In this sense, *recall* is the number of correct matches with respect to the number of corresponding regions while $1 - \text{precision}$ is the number of false matches computed with respect to the total number of regions. Thus, the idea is to increase the correct matches while minimizing the number of false matches. However, previous evaluations illustrate graphically the performance of local descriptors through recall vs 1-precision plots; while, framing the problem in optimization terms requires a quantitative measure that is approached here through the F-measure, see [12]. According to Perez and Olague [1], the F-measure provides a measure that is simple, reliable and objective in the estimation of the local descriptor performance. Hence, this work uses the F_α -Measure with $\alpha = 1$ that refers when precision and recall are evenly weighted; this metric is described as follows:

$$F_\alpha = \frac{(\alpha + 1) \cdot p \cdot r}{(\alpha \cdot p + r)}. \quad (1)$$

where p is precision $\{p : 0 \leq p \leq 1\}$, r is recall $\{r : 0 \leq r \leq 1\}$, and α is a weight for precision or recall, $\{\alpha : 0 \leq \alpha \leq \infty\}$.

3 Synthesis of Local Descriptors' Operators with GP

Genetic programming (GP) is an automatic programming search approach that simulates Darwin's theory of biological evolution to find computer programs that solves a user defined task. The theme of GP to evolve descriptor operators discussed in this paper, offers novel ideas that surprisingly outperforms all previous human-made designs as we can observe in the experiments. Therefore, GP is able to create novel mathematical expressions that are used within the well-known SIFT process with great results and most important, this idea could be potentially extended to other local descriptors. The search space is defined according to previous analysis, see Trujillo and Olague [4]. The function and terminal sets used in our evolutionary algorithm are the following:

$$\begin{aligned} FuncSet &= \left\{ +, |+, |-, |-, *, \div, \sqrt{I_t}, \frac{I_t}{2}, \log_2(I_t), D_x G_\sigma, D_y G_\sigma, G_\sigma \right\} \\ TermSet &= \{I, D_x, D_{xx}, D_{yy}, D_{xy}, D_y\} \end{aligned} \quad (2)$$

where I is the input image and I_t can be any of the terminals in $TermSet$, as well as the output of any of the functions in $FuncSet$; D_u symbolizes the image derivatives along direction u then $D_u = I * G_{u(\sigma=1)}$; G_σ are the Gaussian smoothing filters with $\sigma = 1$ or 2 ; $D_u G_\sigma$ represents the derivative of a Gaussian filter with image blur σ ; $|\gamma|$ is the absolute sum or absolute difference. Moreover, an appropriate fitness function is decisive to the GP process success. Thus, our fitness function is based on a well balanced precision (p) and recall (r) data as described earlier:

$$Q = \operatorname{argmax} \left\{ F(P^x, R^x) = \sum_{i=1}^n \frac{2 \cdot (p_i \cdot r_i)}{p_i + r_i} \right\} \quad (3)$$

with $n = 20$ which represents twenty F-Measure values computed a different descriptor distances for the matching process. Precision data from an image pair are denoted by $P^x = (p_1, p_2, \dots, p_n)$ and recall data by $R^x = (r_1, r_2, \dots, r_n)$. The ranking order of Q is ascendent being $Q = 20$ the highest value that the optimized descriptor operator could obtain; such value corresponds to the descriptor with the best performance.

4 Experimental Results

In this section, we present the experimental results of evolving local descriptors' operators according to the standard testbed provided in [2] following our GP methodology. This testbed is particularly suitable for our purpose of evolving a descriptor operator regardless of the problems of incorporating 3D appearance variation. Indeed, artificial image transformations are used over natural images that are nearly planar and specific camera motions that are approximated as homographies. Therefore, the implementation of our local descriptor was programmed on Matlab with the GP toolbox GPLAB¹. Moreover, the core

¹ <http://gplab.sourceforge.net/index.html>

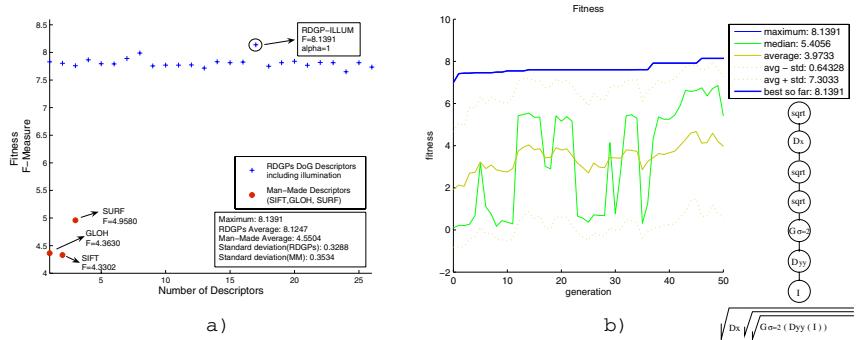


Fig. 1. a) Fitness plot of the 25 trained descriptor operators and 3 human-made descriptors; b) RDGP-ILLUM's tree representation and convergence statistics

platform of our algorithm is based on SIFT features, programmed in Matlab/C². In the time of writing the paper of Perez and Olague [1], we had two set of experiments: 1) GP trained descriptors' operators using the boat image pair and 2) GP trained descriptors' operators using boat + leuven image pairs, see Figure 2. Hence, the first set was published in [1] while the second set is described in this paper.

The training process of our GP learning system takes about 24 hours for obtaining one best descriptor operator while it takes about 6 seconds for testing any of the best evolved operators. The general GP parameters used for training were: 50 generations, 50 individuals, ramped half-and-half as the initialization method, crossover=0.90, mutation=0.10, selection=SUS (Stochastic Universal Sampling) and elitism=0.02. Thus, we obtained 25 best descriptor operators that include illumination information where the best operator is called RDGP-ILLUM. The motivation to include illumination information in the training stage is because the SIFT algorithm is sensitive to this photometric transformation. Figure 1a illustrates a huge difference between the 25 evolved descriptor operators and those state-of-art human-made descriptors; for instance, it is clear to observe that any of the worse evolved descriptors would be better than any of the human-made descriptors. In this sense, the RDGP-ILLUM descriptor is better about 39.09% with respect to the SURF descriptor which is considered as a one of the fastest descriptor algorithms used for real world applications, see [7]. Moreover, Figure 1b shows the RDGP-ILLUM convergence as well as its tree and algebraic representation. On the other hand, we tested the performance of the RDGP-ILLUM, RDGP, SIFT, GLOH and SURF descriptors using the dataset presented in Figure 2. The GLOH descriptor was proposed by Mikolajczyk and Schmid [2] as a SIFT variant; they used polar coordinates instead of cartesian coordinates of the SIFT 3D histogram. Hence, Figure 2 shows the performance of these five descriptors tested with images that present artificial transformations such as: rotation+scale, illumination, blur and JPEG compression. This figure

² <http://vision.ucla.edu/vedaldi/code/sift/sift.html>

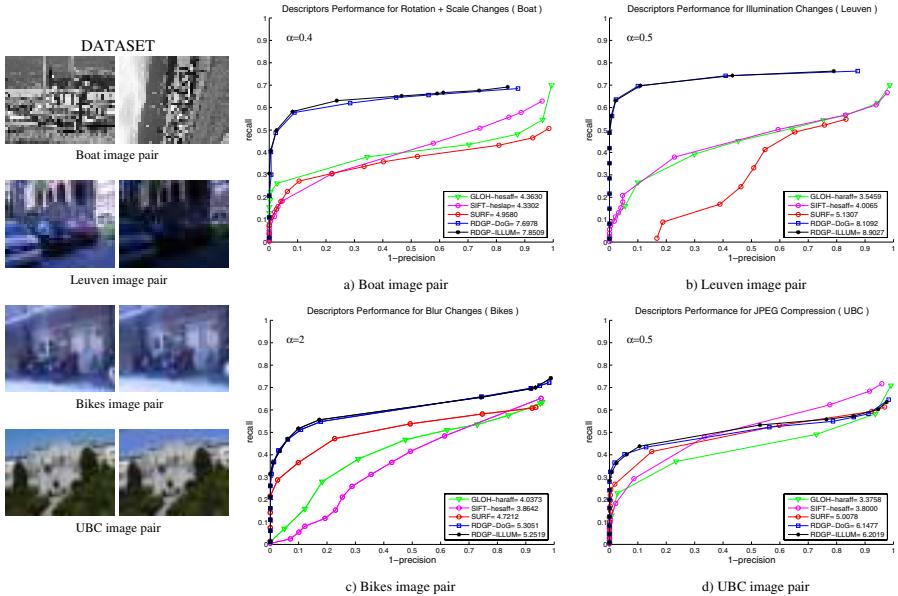


Fig. 2. Datasets used for training and testing and four graphs of the performance evaluation of SIFT, GLOH, SURF, RDGP and RDGP-ILLUM descriptors using different image transformations

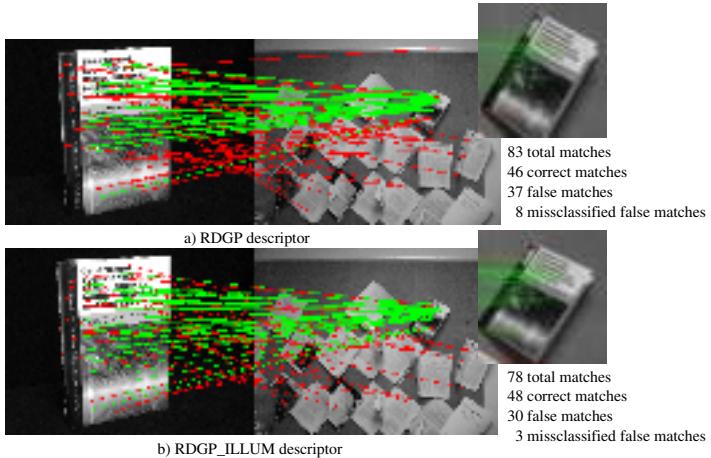


Fig. 3. Book recognition using RDGP and RDGP-ILLUM features

shows that RDGP-ILLUM is better for rotation+scale, illumination and JPEG compression. However, the RDGP is slightly better than RDGP-ILLUM in the case of blur images. In general, the performance of the RDGP-ILLUM descriptor is as good as the RDGP descriptor for different image transformations. Finally,

we used the RDGP-ILLUM descriptor and RDGP descriptor for object recognition application. Hence, Figure 3 shows an example of a book recognition where the green lines represent the correct matches produced in the recognition process while the red lines represent the false matches. Thus, RDGP-ILLUM reduces the number of false matches about 6.11% with respect to the RDGP descriptor.

5 Conclusions

In this paper, we have described a new methodology based on GP with the goal of synthesizing region descriptor operators that actually improves significantly the local descriptor stage. In particular, we have shown in the experimental results that including illumination information during the training process, the resulting descriptor outperforms the RDGP and three state-of-art descriptors. Moreover, RDGP-ILLUM and RDGP descriptors were applied in a real-world recognition problem where RDGP-ILLUM reduced more false features in the matching process.

References

1. Perez, C.B., Olague, G.: Learning Invariant Region Descriptor Operators with Genetic Programming and the F-measure. In: ICPR, December 8–11 (2008)
2. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(10), 1615–1630 (2005)
3. Lowe, D.: Object recognition from local scale-invariant features. *ICCV* 2, 1150–1157 (1999)
4. Trujillo, L., Olague, G.: Using evolution to learn how to perform interest point detection. *ICPR* 1, 211–214 (2006)
5. Trujillo, L., Olague, G.: Synthesis of interest point detectors through genetic programming. In: GECCO, vol. 1, pp. 887–894 (2006)
6. Ke, Y., Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptors. In: CVPR, vol. 2, pp. 506–513 (2004)
7. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006*. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893 (2005)
9. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: CVPR, pp. 1–8 (2008)
10. Winder, S.A.J., Brown, M.: Learning Local Image Descriptors. In: CVPR, pp. 1–8 (2007)
11. Agarwal, S., Roth, D.: Learning a sparse representation for object detection. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) *ECCV 2002*. LNCS, vol. 2353, pp. 97–101. Springer, Heidelberg (2002)
12. Van-Rijsbergen, C.: Information Retrieval. Ed. Butterworth-Heinemann (1979)

Evolutionary Optimization for Plasmon-Assisted Lithography

Caroline Prodhon¹, Demetrio Macías², Farouk Yalaoui¹, Alexandre Vial²,
and Lionel Amodeo¹

¹ Laboratoire d'Optimisation des Systèmes Industriels

² Laboratoire de Nanotechnologie et d'Instrumentation Optique

Institut Charles Delaunay, Université de Technologie de Troyes – CNRS FRE 2848

12, rue Marie Curie BP-2060 F-10010 TROYES Cedex

{caroline.prodhon,demetrio.macias,farouk.yalaoui,
alexandre.vial,lionel.amodeo}@utt.fr

Abstract. We show, through an example in surface-plasmons assisted nano-lithography, the great influence of the definition of the objective function on the quality of the solutions obtained after optimization. We define the visibility and the contrast of a surface-plasmons interference pattern as possible objective functions that will serve to characterize the geometry of a nano-structure. We optimize them with an Elitist Evolution Strategy and compare, by means of some numerical experiments, their effects on the geometrical parameters found. The maximization of the contrast seems to provide solutions more stable than those obtained when the visibility is maximized. Also, it seems to avoid the lack-of-uniqueness problems resulting from the optimization of the visibility.

1 Introduction

Over the past decades, an important amount of work has been devoted to the application of non-classical optimization techniques for the solution of different kinds of inverse and optimization problems in several scientific disciplines [1,2,3]. The nanotechnologies, a fairly new branch of physics, are not an exception and reported results on the use of this kind of methods are becoming more and more common [4,5,6].

Recently, we have employed an Elitist Evolution Strategy to optimally synthesize a nano-structure for an application in plasmons-assisted nanolithography [5]. In that contribution, the objective function to maximize was the visibility of a surface-plasmons interference pattern. For this, we simultaneously optimized the illumination conditions and the geometry of the related nano-structure. In spite of the encouraging results, the definitions of the visibility and the search space were not a sufficient condition to obtain a satisfactory realistic solution, that seems to require the simultaneous fulfillment of two apparently conflicting objectives. In this work, we further explore this issue aiming to have a better understanding on the influence of the definition of the objective function on the solution of problem described in [5].

The structure of this paper is as follows: In Section 2 we formulate the problem and we briefly discuss the difficulties for its solution. Section 3 is devoted to the description of the operational principles of the technique employed for the optimization of a nanostructure. Through some numerical experiments, we illustrate and discuss the performance of our approach in Section 4. We give our main conclusions and final remarks in Section 5.

2 Formulation of the Problem

The geometry considered in the present work is the multilayered system depicted in Fig. 1. The region $x_3 > \zeta_1(x_1)$ is assumed to be a homogeneous isotropic dielectric with constant $\epsilon_0 = 2.22$. The region $\zeta_1(x_1) > x_3 > -t + \zeta_2(x_1)$ is a thin film of silver characterized by its complex dielectric constant $\epsilon_m(\omega) = 1.18 + i0.37$ and a mean thickness t . With reference in Fig. 1, we will assume the surface $\zeta_1(x_1)$ to be plane, whereas the roughness $\zeta_2(x_1)$ in the lower interface of the film is assumed to be a square grating of period arbitrarily fixed to $\Lambda = \frac{\lambda}{10}$. The height and the width of each groove are h_1 and w , respectively. Also, h_2 is the distance between the plane surface $\zeta_1(x_1)$ and the beginning of the groove. The semi-infinite region $x_3 < -t + \zeta_2(x_1)$ is filled with an isotropic media characterized by its dielectric constant $\epsilon_s = 2.25$. For simplicity, the geometry is assumed to be invariant along the direction x_2 and the plane of incidence of the electromagnetic field is the x_1x_3 -plane.

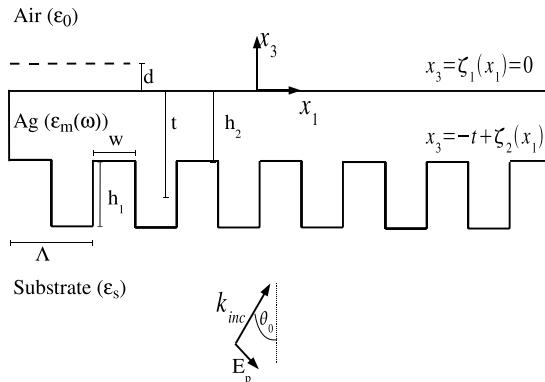


Fig. 1. Metallic grating deposited on a dielectric substrate (see text for details)

The system in Fig. 1 is illuminated from the region $-t + \zeta_2(x_1) > x_3$ with a p -polarized monochromatic plane wave of wavelength $\lambda = 532$ nm at an angle of incidence θ_0 . The near-field scattered intensity $I(x_1)$ is measured in the region $x_3 > \zeta_1(x_1)$ along a line at a constant height $d = 15$ nm from the plane surface $\zeta_1(x_1) = 0$ and it is written as

$$I^{(p,s)}(x_1) = |E^{(p,s)}(x_1)|^2, \quad (1)$$

where $E^{(p,s)}(x_1)$ is the electric field scattered in the near-field of the object.

Although the near-field scattered intensity in Eq. (1) can be obtained by means of different approximated, rigorous or numerical techniques, throughout this work we use the approach based on the Finite-Differences Time-Domain Method (FDTD) described in [5].

The interaction of the system in Fig. 1 with an incident field allows, under certain illumination conditions, the generation surface-plasmons [7]. These electromagnetic surface-waves are significantly sensitive to the geometrical and material characteristics of their surrounding environment and this fact has led to different applications as biosensing [8] or plasmon-assisted nanolithography [9], to name but a few examples. For the numerical experiments conducted in [5], we defined the visibility of the surface-plasmons interference pattern localized on the surface $\zeta_1(x_1)$ of Fig.1 as

$$V(x_1|\mathbf{p}) = \frac{I^{(p)}(x_1|\mathbf{p})_{max} - I^{(p)}(x_1|\mathbf{p})_{min}}{I^{(p)}(x_1|\mathbf{p})_{max} + I^{(p)}(x_1|\mathbf{p})_{min}}, \quad (2)$$

where $\mathbf{p}(\theta_0, h_1, h_2, w)$ is the vector of objective variables to be determined and $I^{(p,s)}(x_1)_{max}$ and $I^{(p,s)}(x_1)_{min}$ are the maximal and minimal intensity levels of the resulting interference pattern, respectively. Although we obtained feasible solutions providing a maximal visibility $V \approx 1$, in most of the numerical experiments the contrast

$$C(x_1|\mathbf{p}) = I^{(p)}(x_1|\mathbf{p})_{max} - I^{(p)}(x_1|\mathbf{p})_{min} \quad (3)$$

was weak. This situation contradicted the hypothesis that it is possible to tailor, in an optimal and controlled manner, through the maximization of amplitude of the interference fringes, the topography of a photosensitive material film deposited on the region $x_3 > \zeta_1(x_1)$ of Fig. 1 [9].

From the previous discussion, it seems that maximizing the contrast $C(x_1|\mathbf{p})$ rather than the visibility $V(x_1|\mathbf{p})$ should provide more information about the phenomena involved in the generation of the interference pattern.

3 The Optimization Algorithm

Although its operational principles can be found elsewhere [5], we consider convenient to briefly describe the optimization technique employed in this work.

3.1 Representation of the Objective Variables

Usually, the scalar magnitudes involved in an experimental set up are defined within the real domain $\mathbf{R}^{(1)}$. In addition, independently of the approach employed for modeling the phenomenon under study, the numerical solution of the related scattering equations is also done in $\mathbf{R}^{(1)}$. It seems thus natural to make use of a real representation scheme for the parameters $\mathbf{p}(\theta_0, h_1, h_2, w)$.

3.2 Evolution Strategies

In principle, any of the optimization techniques reported in the literature could be employed to maximize Eqs. (2) and (3). However, as discussed in [5], their form and the constraint imposed by the representation scheme suggest the Evolution Strategies as a suitable option to solve the problem studied in this work.

The starting point of the optimization process is the generation of a random set $P_\mu^{\langle g \rangle}|_{g=0}$ of μ possible solutions to the problem, which in the present context are the objective variables $\mathbf{p}(\theta_0, h_1, h_2, w)$ (see Fig. 1). A secondary population $P_\lambda^{\langle g \rangle}$ of λ elements is then generated through the application of the recombination and mutation operators over the elements of the initial population $P_\mu^{\langle g \rangle}$. It is pertinent to mention that the index g stands for a given iteration of the evolutionary loop.

Recombination exploits the search space through the exchange of information between ρ different elements of the population. Mutation, on the other hand, explores the search space through the introduction of random variations in the newly recombined elements. Also, depending on the problem studied, it is possible to exclude the recombination operation from the evolutionary loop.

Once the secondary population $P_\lambda^{\langle g \rangle}$ has been generated, one needs to evaluate the quality of its elements. For this, the direct problem must be solved for each one of the newly generated elements of the secondary population. With this, a fitness value is associated to each trial system. In the present context, this is done through the evaluation of the contrast on the basis of Eq. (3). Only those elements of the secondary population $P_\lambda^{\langle g \rangle}$ leading to promising regions of the search space will be retained, through some selection scheme, as part of the population $P_\mu^{\langle g+1 \rangle}$ for the next iteration of the evolutionary loop. The procedure is repeated until a defined termination criterion has been achieved. The respective sizes of the initial and the secondary populations, $P_\mu^{\langle g \rangle}$ and $P_\lambda^{\langle g \rangle}$, remain constant throughout the entire search process.

4 Results and Discussion

To illustrate the optimization technique just described, we consider the system depicted in Fig. 1 under the illumination and material conditions established in Sect. 2. Our goal is to maximize the contrast $C(x_1|\mathbf{p})$ (Eq. 3) through the determination of the optical and geometrical parameters $\mathbf{p}(\theta_0, h_1, h_2, w)$ involved in the generation of the surface-plasmons interference pattern.

We employ a self-adaptive Elitist ($ES - (\mu/\rho + \lambda)$) evolution strategy with $\mu = 14$, $\lambda = 100$ and $\rho = 2$ elements to be recombined [5]. The maximum number of iterations is set to $g_{max} = 30$ and it provides the termination criterion for the search process. Each individual in the initial and secondary populations is represented by the components of the vector \mathbf{p} . To reduce the search space and to avoid the unnecessary evaluation of non-physical solutions, we set the upper limits $\theta_{0(max)} = 80^\circ$ and $h_{2(max)} = 0.25\lambda_0$ for greatest angle of incidence and the largest distance h_2 , respectively. Also, the largest height h_1 and width w of each

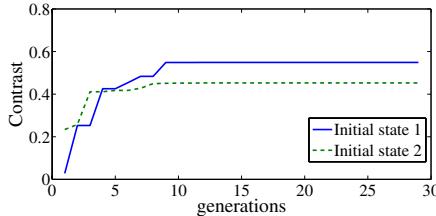


Fig. 2. Convergence behavior of the contrast

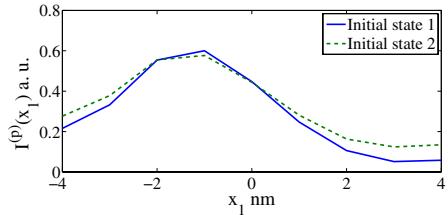


Fig. 3. Interference pattern generated with the contrast parameters in Table 1

Table 1. Parameters synthesized with the algorithm proposed

Objective function: Visibility						
Initial state	θ_0 [°]	h_2 [nm]	h_1 [nm]	w [nm]	V	C^*
1	45	19	85	19	≈ 1	1.38×10^{-3}
2	53	19	41	26	≈ 1	3.42×10^{-3}

Objective function: Contrast						
Initial state	θ_0 [°]	h_2 [nm]	h_1 [nm]	w [nm]	C	V^*
1	35	14	51	48	0.54	0.84
2	34	19	52	40	0.45	0.64

*Values estimated analytically.

groove of the grating are respectively set to $h_{1(max)} = 0.25\lambda_0$ and $w_{max} = 0.9\Lambda$. Moreover, we define the components of the vector \mathbf{s} of strategy parameters, associated to the mutation procedure, as $\sigma_{\theta_0} = 0.2\theta_{0(max)}$, $\sigma_{h_1} = 0.2h_{1(max)}$, $\sigma_{h_2} = 0.2h_{2(max)}$ and $\sigma_{w_{max}} = 0.2w_{max}$.

For a fair comparison, the optimization procedure was tested for its relative success by searching for the solution starting from the same 30 randomly generated initial states employed in [5]. Each of them consisted of μ vectors \mathbf{p} and their respective associated strategy-parameters vectors \mathbf{s} .

In Fig. 2 we present the typical monotonic convergence behavior of the Elitist Strategy, for two initial states selected among the 30 ones considered. The curves in Fig. 3 correspond to one period of the surface-plasmons interference pattern obtained after optimization of the contrast. Despite the slight differences, the maximal amplitudes of the two fringes are about the same value. This result suggests that optimizing the contrast rather than the visibility avoids the lack of uniqueness of the solution, as shown in Table 1, where the contrasts associated to optimal visibilities close to the unit are significantly different. Moreover, the parameters associated to the optimal contrast do not exhibit discrepancies as important as those present when the visibility is maximized. Thus, the optimization of the contrast seems to provide solutions closer to the physical situation considered within the context of the present work.

5 Conclusions and Final Remarks

The significant effect of the objective function on the optimization process was illustrated through an example in surface-plasmons nano-lithography. The ideal situation in which the contrast and the visibility are maximal does not seem attainable when these two objective functions are optimized independently. However, it is noteworthy the remarkable stability of the solutions found through the maximization of the contrast. Although we are not aware of many systematic studies using Stochastic Optimization within the context of surface-plasmons nano-lithography and the results presented here are promising, they are far from being conclusive and further work is required. A comparison of the results obtained using different optimization techniques and also different definitions of the visibility should provide a better understanding on the behavior of the solutions obtained. Currently, work is in progress towards the simultaneous optimization of the visibility and the contrast through the Pareto Archived Evolution Strategy (PAES).

References

1. Hodgson, R.J.W.: Genetic Algorithm Approach to Particle Identification by Light Scattering. *Journal of Colloid and Interface Science* 229, 399–406 (2000)
2. Prins, C., Prodhon, C., Soriano, P., Ruiz, A., Wolfier-Calvo, R.: Solving the Capacitated LRP by a Cooperative Lagrangean Relaxation-Granular Tabu Search Heuristic. *Transportation Science* 41, 470–483 (2007)
3. Macías, D., Olague, G., Méndez, E.R.: Inverse scattering with far-field intensity data: random surfaces that belong to a well-defined statistical class. *Waves in Random and Complex Media* 16, 545–560 (2006)
4. Kildishev, A.V., Chettiar, U.K., Liu, Z., Shalaev, V.M., Kwon, D., Bayraktar, Z., Werner, D.H.: Stochastic optimization of low-loss optical negative-index metamaterial. *J. Opt. Soc. Am. B* 24, A34–A39 (2007)
5. Macías, D., Vial, A.: Optimal design of plasmonic nanostructures for plasmon-interference assisted lithography. *Appl. Phys. B* 93, 159–163 (2008)
6. Barchiesi, D., Macías, D., Belmar-Letellier, L., van Labeke, D., Lamy de la Chapelle, M., Toury, T., Kremer, E., Moreau, L., Grosge, T.: Plasmonics: influence of the intermediate (or stick) layer on the efficiency of sensors. *Appl. Phys. B* 93, 177–181 (2008)
7. Maier, S.A.: *Plasmonics: Fundamentals and Applications*, p. 223. Springer, Heidelberg (2007)
8. Homola, J.: *Surface Plasmon Resonance Based Sensors*, p. 251. Springer, Berlin (2006)
9. Derouard, M., Hazart, J., Lérondel, G., Bachélot, R., Adam, P.-M., Royer, P.: Polarization-sensitive printing of surface plasmon interferences. *Optics Express* 15, 4238–4246 (2007)

An Improved Multi-objective Technique for Fuzzy Clustering with Application to IRS Image Segmentation

Indrajit Saha¹, Ujjwal Maulik², and Sanghamitra Bandyopadhyay³

¹ Department of Information Technology, Academy of Technology,
Adisaptagram-712121, West Bengal, India
indra_raju@yahoo.co.in

² Department of Computer Science and Engineering, Jadavpur University,
Jadavpur-700032, West Bengal, India
drumaulik@cse.jdvu.ac.in

³ Machine Intelligence Unit, Indian Statistical Institute, Kolkata-700108,
West Bengal, India
sanghami@isical.ac.in

Abstract. In this article a multiobjective technique using improved differential evolution for fuzzy clustering has been proposed that optimizes multiple validity measures simultaneously. The resultant set of near-pareto-optimal solutions contains a number of nondominated solutions, which the user can judge relatively and pick up the most promising one according to the problem requirements. Real-coded encoding of the cluster centres is used for this purpose. Results demonstrating the effectiveness of the proposed technique are provided for numeric remote sensing data described in terms of feature vectors. One satellite image has also been classified using the proposed technique to establish its efficiency.

Keywords: Fuzzy clustering, improved differential evolution, multiobjective optimization, pareto-optimal.

1 Introduction

Clustering is a useful unsupervised data mining technique which partitions the input space into K regions depending on some similarity/dissimilarity metric where the value of K may or may not be known *a priori*. The main objective of any clustering technique is to produce a $K \times n$ partition matrix $U(X)$ of the given data set X , consisting of n patterns, $X = \{x_1, x_2, \dots, x_n\}$. The partition matrix may be represented as $U = [u_{k,j}]$, $k = 1, \dots, K$ and $j = 1, \dots, n$, where $u_{k,j}$ is the membership of pattern x_j to the k th cluster.

In 1995 a new floating point encoded evolutionary algorithm for global optimization called Differential Evolution (DE) [1] was proposed that uses a special kind of differential operator. The improved variant of differential evolution differs from the classical differential evolution in the process of mutation. While doing the mutation it uses three vectors, one representing the local best and

the other the global best which are adaptive in nature and third one is selected randomly.

In this article, the efficiency of the proposed algorithm as compared with the well known Fuzzy C-Means (FCM) [2] (single objective optimization algorithm), NSGA-II based fuzzy clustering algorithm [3] and also a multiobjective version of classical differential evolution based fuzzy clustering technique (MOIDEFC) are also provided.

2 Improved Differential Evolution

In Improved Differential Evolution (IDE) an approach has been introduced during mutation to push the trial vector quickly towards the global optima. The crossover and selection are same as original DE. In the mutation process, that deals with three vectors, two of them represent the global best (GBest) and local best (LBest) respectively. In each generation α (alpha) is computed as $\frac{1}{1+exp(-(1/generation))}$ which uses as a mutation factor and puts an effect on new mutant vector, created by Eqn. 1. Note that as number of generations increases the value of α decreases in the range between [1, 0.5].

$$\vartheta_{i,l}(t+1) = G_{GBest,l}(t) + \alpha (G_{LBest,l}(t) - G_{m,l}(t)) \quad (1)$$

3 Proposed Multiobjective Improved Differential Evolution Based Fuzzy Clustering

In this section, we describe the proposed multiobjective technique MOIDEFC for evolving a set of near-pareto-optimal non-degenerate fuzzy partition matrices.

3.1 Vector Representation and Initial Population

In the MOIDE based fuzzy clustering, the vectors are made up of real numbers which represent the coordinates of the centres of the partitions. If there are K clusters in the data set, the centres encoded in a vector in the initial population are randomly chosen among K distinct points from the data set.

3.2 Computation of the Objectives

In MOIDEFC the objectives associated with each vector. Performance of the multiobjective clustering highly depends on the choice of objectives which should be as contradictory as possible. In this article, the XB [4] and J_m [2] have been chosen as the two objectives to be optimized. It can be noted that J_m calculates the global cluster variance. Thus lower value of J_m implies better clustering solution. On the other hand, the XB index is a combination of global and local situations. Considering J_m and XB will provide a set of alternate partitioning of the data. Hence it is evident that both J_m and XB indices are needed to be minimized in order to get good solutions. The different steps of MOIDE are shown in Fig. 1.

```

1. Initialize the vectors of the population.
2. Evaluate objective values of each parent vector.
3. Perform nondominated sorting to assign rank.
4. Compute the crowding distance between members of each front.
5. Set GBEST and LBEST (The solution vector of lowest rank with least
   crowding distance).
Repeat
  alpha=1/(1+exp(-(1/generation)))
  7. modiMutation
  8. Crossover
  9. Evaluate objective value of each offspring vector.
  10. Combine parent vectors with offspring vectors to create new
      population.
  11. Perform Non-dominated sorting to assign rank.
  12. Compute the crowding distance between members of each front.
  13. Update LBEST (The solution vector of lowest rank with least
      crowding distance).
  if(LBEST<GBEST)
    14. Set GBEST with LBEST
  endif
  15. Select the vectors of the combined population based on
      nondominated lowest rank vectors of size same as population of
      the next generation. (Elitism)
Until (termination criteria are met)

```

Fig. 1. MOIDE algorithm

4 Experimental Results

4.1 Numerical Remote Sensing Data

SPOT: This data set [5] consisting of 932 samples into seven distinct classes.

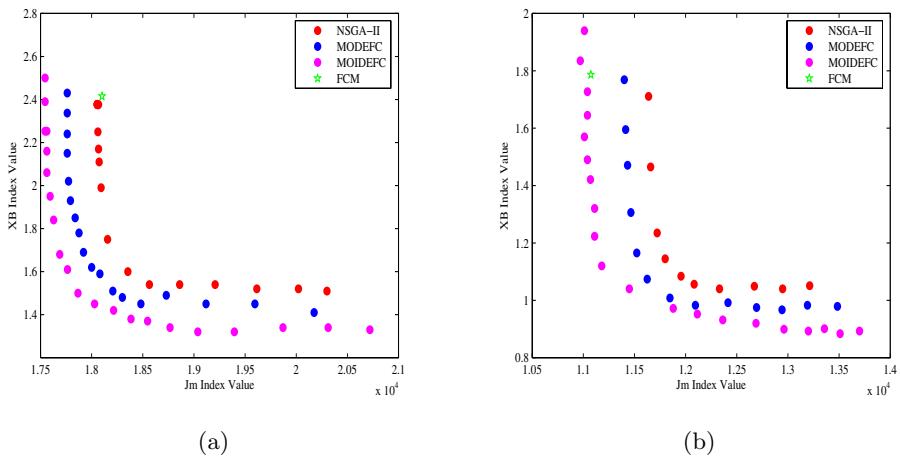
LANDSAT: This data set has 795 samples [5]. The data set contains five classes.

4.2 Input Parameters

The population size and number of generation used for MOIDEFC, MODEFC and NSGA-II based fuzzy clustering algorithms are 50 and 100 respectively. The crossover probability and mutation probability for NSGA-II based fuzzy clustering are 0.8 and $(1/\text{length of chromosome})$ respectively. The mutation factors (F and α) for MODEFC and MOIDEFC algorithms are set to 0.8 and $\frac{1}{1+\exp(-(1/generation))}$ respectively. The crossover probability for MOIDEFC, MODEFC, is taken as 0.8. The results provided corresponding to the proposed multiobjective scheme is obtained by selecting the solution of the final non-dominated front that provides the best Minkowski Score (MS) [6] value. For all the fuzzy clustering algorithms m , the fuzzy exponent, is set to 2.0.

Table 1. Average J_m , XB, MS and \mathcal{I} indices for different Data Set

Data Set	Algorithm	J_m	XB	MS	\mathcal{I}
SPOT	MOIDEFC	18214.0573	1.4731	0.7141	2566.0924
	MODEFC	18481.0425	1.4795	0.8031	2453.7253
	NSGA-II	18564.7362	1.5482	0.8831	2315.9253
	FCM	18102.9351	2.4152	0.9562	2273.8169
LANDSAT	MOIDEFC	11180.6372	1.1272	0.5855	410.9572
	MODEFC	11522.1652	1.2689	0.6714	367.4381
	NSGA-II	11658.0746	1.4638	0.7513	296.7461
	FCM	11074.4462	1.7807	0.8251	259.2027

**Fig. 2.** Best Nondominated Pareto front for (a) SPOT and (b) LANDSAT data set

4.3 Results

Table 1 shows the comparative results are obtained for the two data sets average over 50 runs. It is seen from the table that the proposed method consistently outperforms the MODEFC, NSGA-II as well as FCM algorithm in terms of the MS and \mathcal{I} indices values. For example, for SPOT, the proposed multiobjective technique achieves an XB value of 1.4731 while the MODEFC and NSAGA-II provide values of 1.4795 and 1.5482 respectively. Similar results are found for the other data set. Fig. 2 demonstrates the best nondominated pareto front produced by MOIDEFC, MODEFC and NSGA-II algorithms for all the data sets. Clearly the nondominated pareto front produced by MOIDEFC is better for all the data sets.

5 Application to IRS Image Segmentation

Fig. 3.b, 3.c, 3.d and 3.e shows the IRS Calcutta image clustered using proposed MOIDEFC, MODEFC, NSGA-II and FCM algorithms respectively. From

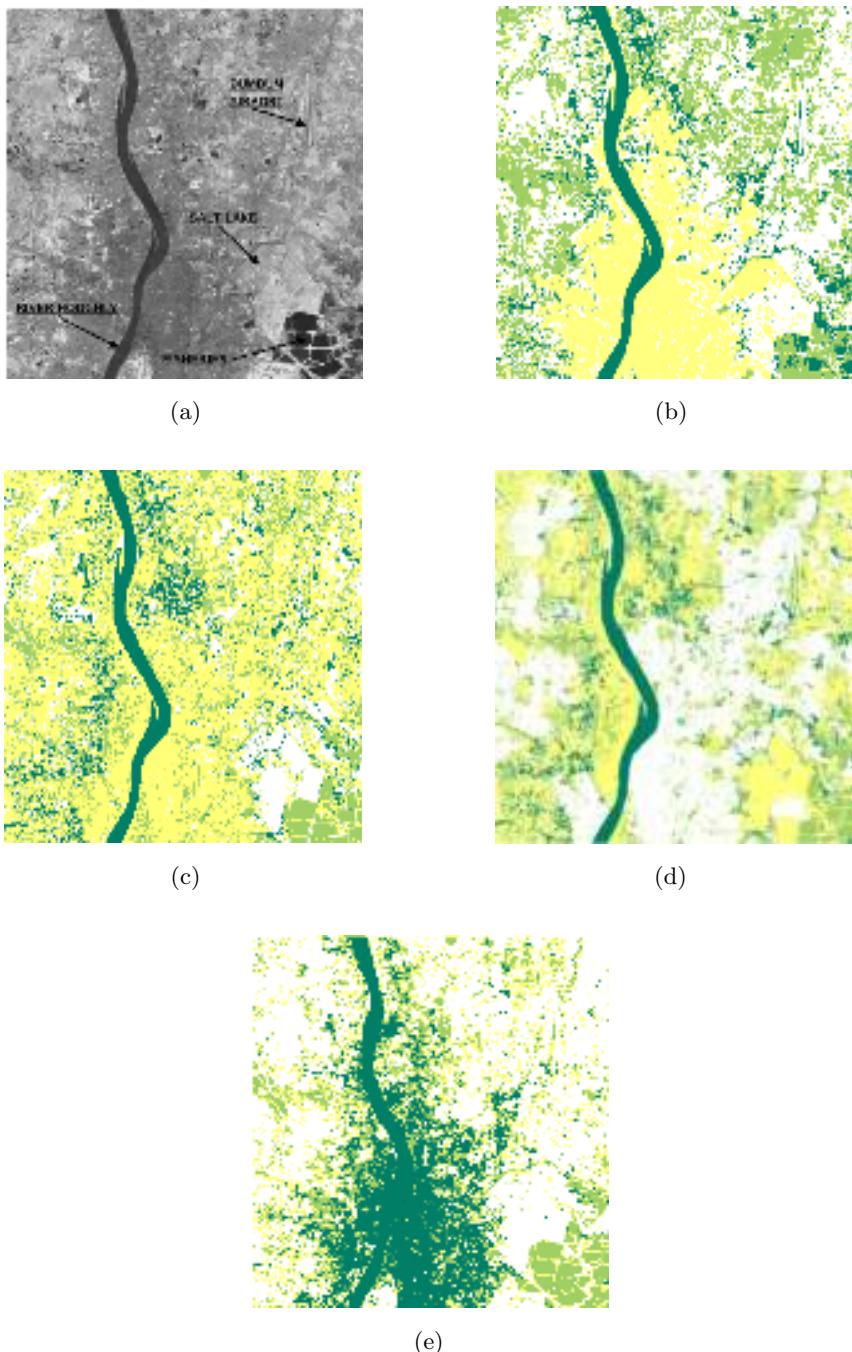


Fig. 3. (a) IRS image of Calcutta in the NIR band with histogram equalization. (b) Clustered IRS image of Calcutta using MOIDEFC. (c) Clustered IRS image of Calcutta using MODEFC. (d) Clustered IRS image of Calcutta using NSGA-II. (e) Clustered IRS image of Calcutta using FCM.

Table 2. Average J_m , XB and \mathcal{I} indices values for IRS Calcutta image

Algorithm	J_m	XB	\mathcal{I}
MOIDEFC	3652467.41	1.6322	118.4101
MODEFC	3652665.82	2.0825	103.9625
NSGA-II	3652803.17	2.1115	88.0750
FCM	3652081.35	2.1409	18.0436

ground knowledge, it is known that the image has four classes [5]: turbidwater (TW determine as deep green), pond water (PW determine as light green), concrete (Concr. determine as yellow) and open space (OS determine as white). It appears from Fig. 3.b very clearly. From the Fig. 3.e, it can be noted that the river Hooghly and the city region. Results produced by MODEFC and NSGA-II are better than FCM but poorer compared to MOIDEFC. This is also evident from the J_m , XB and \mathcal{I} indices reported in Table 2.

6 Conclusion

In this article a new multiobjective improved differential evolution based fuzzy clustering technique has been developed. Results on different numerical remote sensing data sets indicate that MOIDEFC consistently performs better than MODEFC, NSGA-II based fuzzy clustering and FCM algorithms. In this context, an IRS satellite image of Calcutta has been classified using the developed multiobjective technique and compared with other clustering algorithms.

References

1. Storn, R., Price, K.: Differential evolution - A simple and efficient heuristic strategy for global optimization over continuous spaces. *Journal of Global Optimization* 11, 341–359 (1997)
2. Bezdek, J.C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York (1981)
3. Bandyopadhyay, S., Maulik, U., Mukhopadhyay, A.: Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing* 45(5), 1506–1511 (2007)
4. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13, 841–847 (1991)
5. Maulik, U., Bandyopadhyay, S.: Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *IEEE Transactions on Geoscience and Remote Sensing* 41(5), 1075 (2003)
6. Jardine, N., Sibson, R.: *Mathematical Taxonomy*. Wiley, London (1971)

Interactive Evolutionary Evaluation through Spatial Partitioning of Fitness Zones

Namrata Khemka¹, Gerald Hushlak³, and Christian Jacob^{1,2}

¹ Dept. of Computer Science, Faculty of Science

² Dept. of Biochemistry & Molecular Biology, Faculty of Medicine

³ Faculty of Fine Arts

University of Calgary, Calgary, Alberta, Canada

Abstract. This paper discusses how large-scale interactive evolutionary design can be accomplished through innovative evaluation interfaces. An application example from the world of textile designing and fine arts serves to illustrate an evolutionary evaluation interface using spatial arrangements. The new interface allows the designer to drag and drop images that represent solutions into fitness zones on the screen. As our team consists of two computer scientists and an artist, we also explore the collaborative relationships among the team members, and between the artist and the evolutionary system.

1 Introduction

The power of the unaided mind is highly overrated. Without external aids, memory, thought, and reasoning are all constrained. But human intelligence is highly flexible and adaptive, superb at inventing procedures and objects that overcomes its own limits. The real powers come from devising external aids that enhance cognitive abilities. How can we have increased memory, thought, and reasoning? By the inventions of external aids: It is things that make us smart.

Donald Norman, Univ. of California, San Diego

External aids play a key role in human cognition and facilitate an enhanced understanding through exploration of the worlds that surround us, including virtual worlds that become more and more abundant. Computers are one of the most advanced ‘think tools’ we have. In particular, computers are becoming increasingly useful in the wide area of design, whether we construct a new airliner—virtual models are developed and tested long time before an actual physical airplane is built, whether we streamline the shape of a car—performing virtual wind tunnel experiments on a computer, or whether we want to improve on the designs of next-generation computer chips—which are completely simulated *in silico* before the actual chip is manufactured.

As helpful as computers are in accomplishing such complex design tasks, we still tend to engineer our designs, where we assemble sub-modules into larger and

more elaborate systems in an almost brute-force, step-by-step manner. To expedite efficiencies in manufacture, top down design proceeds in the most efficient way towards a successful outcome. The focus on successful outcome deemphasizes the opportunity for play and failures which feed the creative process and invention. Nature relies on a much different way of designing its enormously complex living ‘machines’. The key to Nature’s massively parallel design processes is evolution, where design solutions are tested against each other, whereby many failing designs are created in the process. Over time, successfully adapted solutions rise to the top of the design hierarchy.

Inspired by and based on the concepts of evolution in Nature, evolutionary systems have helped to generate innovative results and have been successfully applied in various domains such as engineering, medicine, architecture, and art [1], [2]. Innovative designers utilize evolutionary algorithms to produce potential solutions for search problems by iteratively exploring design options through mutations and recombinations of previous design ideas. The better solutions survive for evolutionary breeding, while poor solutions disappear from the population of designs. This process is repeated until satisfactory solutions are found (Figure 1).

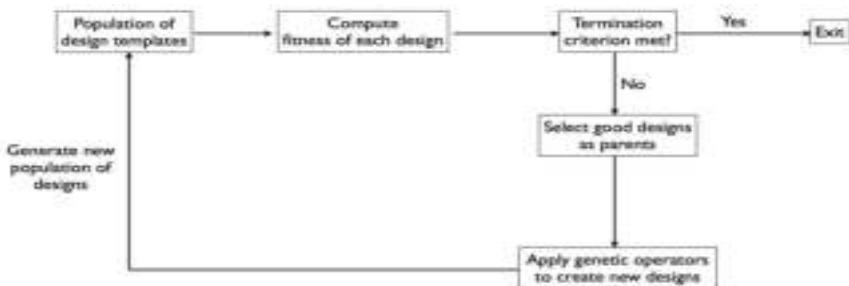


Fig. 1. Basic evolutionary algorithm process

2 Related Work

Interactive Evolutionary Computation (IEC) is used in domains where the fitness function is not known in advance, or where it is difficult to define the criteria to computationally assess good quality solutions—such as visual appeal, aesthetics, and attractiveness. Recursive two-dimensional line figures called Biomorphs, invented by Richard Dawkins, were the first demonstrations of how interactivity can be used for simple parameter evolution [3]. Inspired by Dawkins’ work, Latham and Todd created genetically evolved three-dimensional artificial life-like forms that can be considered as “virtual sculptures” [4]. The works of both Dawkins and Latham and Todd had prompted Karl Sims to generate abstract two-dimensional images based on symbolic expressions for computer art, evolved under the guidance of humans [5]. Sims further inspired many computer scientists to use evolution for generating art. Unemi developed Sbart, an image breeding program using selection to evolve images similar to Sims [6]. Rooke used genetic

algorithms to evolve fractal art constructed from the Mandelbrot set schemes [1]. Takagi compiled an extensive survey of applied interactive evolutionary computation in areas such as entertainment, data mining, music, and animation [7]. Bentley applied a human-guided fitness evaluator for generating shapes such as drums, flowers, and robots [1]. Some of our team members have also gained considerable experience with interactive SwarmArt installations, utilizing both swarm intelligence and audience interactive live video processing [8].

3 Interactive Evolutionary Design with *Inspirica*

Inspired by the capability of evolutionary computation, *Evolvica* was one of the earlier evolutionary design systems from the 1990s utilizing genetic algorithms, genetic programming, evolution strategies, and evolutionary programming. *Inspirica* [9] is an extension of *Evolvica* and supports evolutionary exploration of design solutions through either automatic fitness evaluation or interactive assessment by a ‘breeder’ (Figure 2(a)), following Richard Dawkins’ original idea of evolutionary biomorph design [3]. Over the last few years, *Inspirica* has been applied to the design of implicit surface models of fluid containers, chairs, face masks, and swarm dynamics [10].

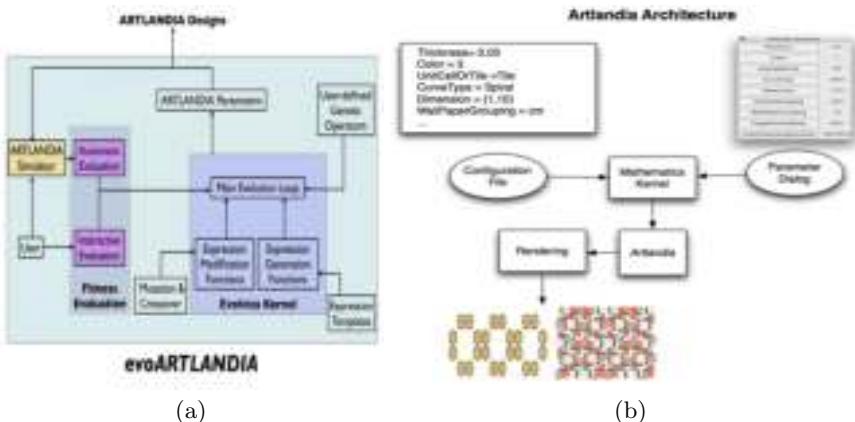


Fig. 2. (a) Overview of the *Inspirica* system used for evolving the parameters of the *evoArtlandia* application. Each box in the diagram is a component within the evolutionary design environment. Arrows indicate the flow of output from one component into another. (b) System architecture of *evoArtlandia*: The parameters are passed to the *Mathematica* kernel via a configuration file or via the parameter dialog. *Artlandia* renders the patterns using *Mathematica*’s graphics library.

The *Inspirica* interface consists of an evaluation window and a phenotype window for each individual in the population. Figure 3 illustrates nine *Artlandia* [11] phenotype windows along with their corresponding evaluation windows. The

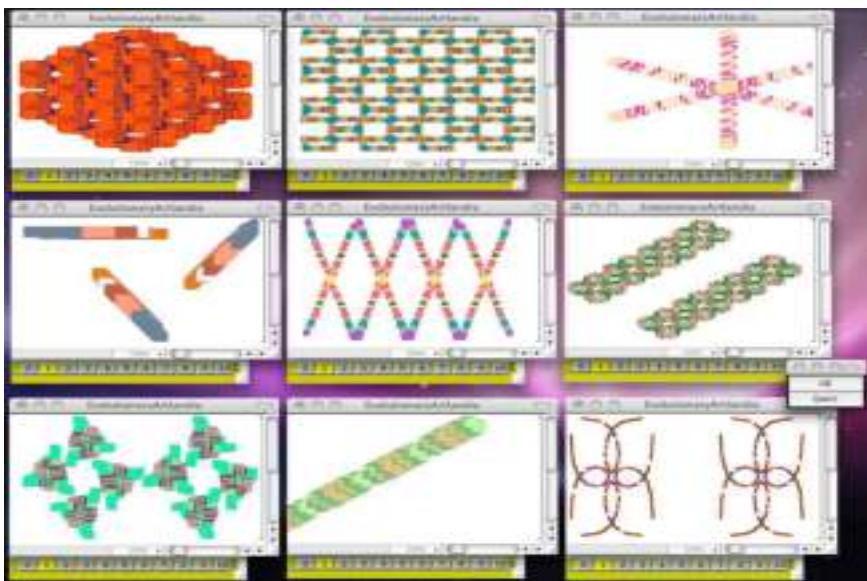


Fig. 3. Interactive *evoArtlandia* user interface: A screen shot of the interactive *Artlandia* user interface is displayed here. The user can steer the evolution by manually assigning fitness values on a scale from 0 to 10 to each individual in the population. By clicking the “OK” button, the user creates the next generation of the tiled patterns. The breeding process is terminated by selecting the “Quit” Button.

evaluation windows allow the user to rank each respective window by assigning a fitness value from 0 to 10 when clicking on the respective buttons. A fitness of 0 removes the individual from the population, thereby reducing the size of the next generation. The user is also presented with a control panel that generates the next generation of individuals or stops the experiment. The following describes the main evolution loop used with *Inspirica* [9]:

1. The main control panel is displayed.
2. The initial population is randomly generated from the templates when the evolution process in *Inspirica* starts. Expression generation templates are provided by the user and depend on the problem domain.
3. The phenotype and evaluation windows are presented to the user.
4. By clicking the “OK” button the next generation is created.
 - The best individual is copied without any change into the new population.
 - A parent is selected through roulette wheel selection based on its fitness.
 - A genetic operator is chosen through roulette wheel selection. The weighting of the mutation and recombination operators is initially defined by the user, and is automatically adjusted during an evolution run.
 - The selected genetic operator is applied to the chosen parent (or two parents in case of crossover) to produce new individuals.

- The new individual is added to the next generation population. In the case of crossover, the two new individuals are added to the new population only if there is enough room. Otherwise, only one of them, chosen at random, is added.
 - This process is repeated until the number of individuals in the new population equals the number of individuals in the parent population.
5. This new population becomes the next generation.
 6. The whole evaluation-selection-mutation process is repeated until the user hits the “Quit” button on the control panel.

4 Designing Tiled *Artlandia* Patterns

We use the *Artlandia* [11] software package to create a variety of mathematically specified two-dimensional forms, applicable to the design of tiling patterns. In *Artlandia* tiling patterns are created by constructing pre-defined curves such as *spiral*, *ellipse*, *parabola*, *cubic parabola*, etc. (Figures 4(a) and 5(a)). *UnitCell* or *Tile* object of a specified symmetry group is created from the base curve (Figures 4(b) and 5(b)). Finally, ‘tiling’ functions (*cm*, *p4*, *p6m*, etc.) are applied to *UnitCell* or *Tile* objects, resulting in a tiled pattern (Figures 4(c) and 5(c)). In total, fifteen parameters influence the creation of the tiling patterns. The *Artlandia* application is written in *Mathematica* [12]. All the rendering is conducted in *Mathematica*. The parameters for rendering are either set by the user via the parameter dialog or through a configuration file as illustrated in Figure 2(b).

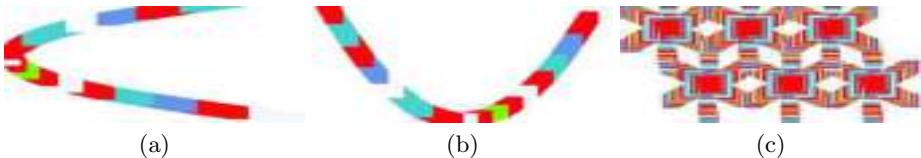


Fig. 4. A typical example of how to generate a tiling pattern using the *UnitCell* object. (a) Hyperbola as the basic construct. (b) A *UnitCell* object oriented along the *YAxis* is constructed. (c) A tiling arrangement is achieved using the *p6* symmetry pattern.

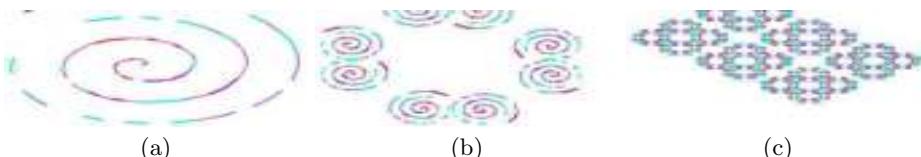


Fig. 5. An example of tiling pattern generation using the *Tile* object. (a) Spiral as the basic construct. (b) A *Tile* object with a *p4g* symmetry is created. (c) A more complex arrangement is created from the *Tile* object building block using the *pmm* symmetry pattern.

5 Challenges and Benefits of Interactive Evolutionary Design

We have used *Inspirica* to evolve tiled patterns. The interactive evaluator in *Inspirica* allows users to steer the evolution by manually assigning a fitness value to each individual (Figure 3). In each experiment, a collection of ‘phenotype windows’ is presented to the user. A big problem arises, though, when the human evaluator is confronted with a large number of designs to be evaluated, such as illustrated in Figure 6. How can a human—who always is the bottleneck in an interactive evolutionary system—efficiently rank the presented solutions? Of course, this has very practical consequences. In many design systems—including evolution-inspired designs—the human expert represents an essential part of the overall search process and is significantly advantaged by being able to subconsciously connect information in an organic approach as opposed to linear succession. It is in many cases the human who has the unique ability to quickly evaluate specific features that the solutions provided by the ‘external aid’ exhibit. If there is a proper way to capture solution fitness by a mathematical formula, then this is not a problem at all. Fitness evaluation can then be implemented by an automatic procedure defined through training. However, if designs involve criteria such as aesthetics, subjective feasibility, general appearance, or anything for which—at least at the start of the design process—one has no clear idea of how to formulate a selection criterion, then the human designer remains

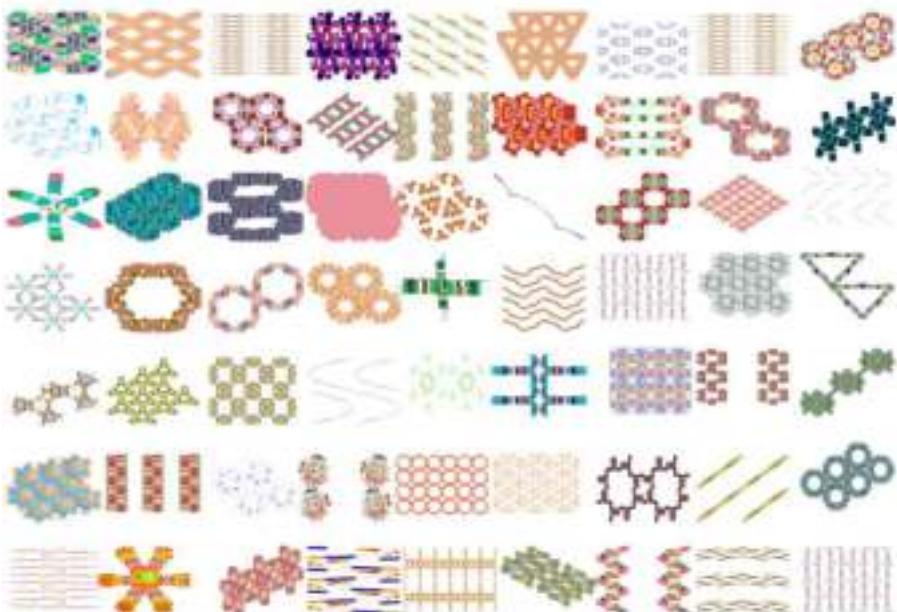


Fig. 6. The interactive evolutionary design dilemma: How can a human designer—the breeder—efficiently evaluate a large number of solutions?

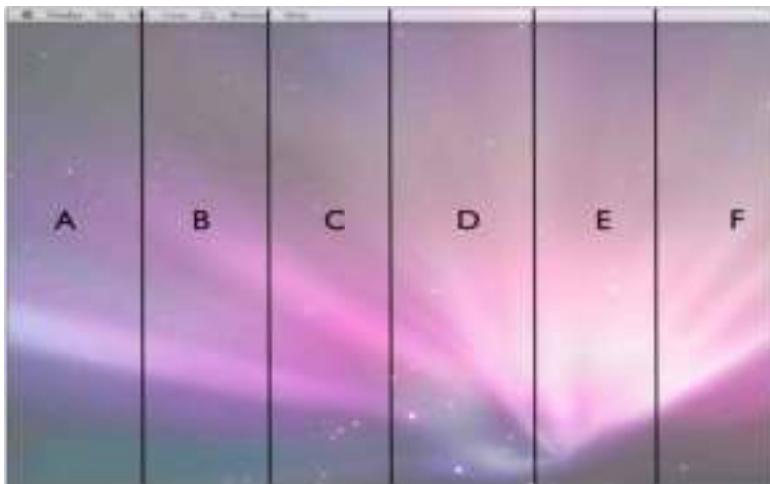


Fig. 7. Overview of the human-centered interface for interactive evolutionary systems. First, the user defines the arrangement of the fitness zones (A-F). Each zone is then associated with a specific fitness value or a range of fitness values.

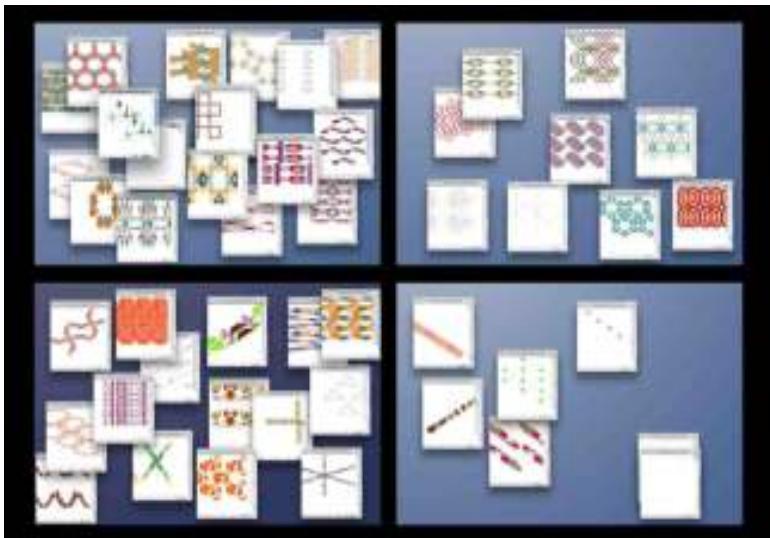


Fig. 8. Fitness assignment through spatial arrangement: 50 phenotype windows arranged in four evaluation zones. Solutions placed in the top right corner have the highest fitness, with fitnesses decreasing counter clock-wise over the quadrants.

a key collaborator, director, and ‘breeder’ within the overall process [2]. In fact, it should be the ultimate goal to turn evolutionary design systems into helpful tools that enhance, accelerate, and improve both the exploratory design process

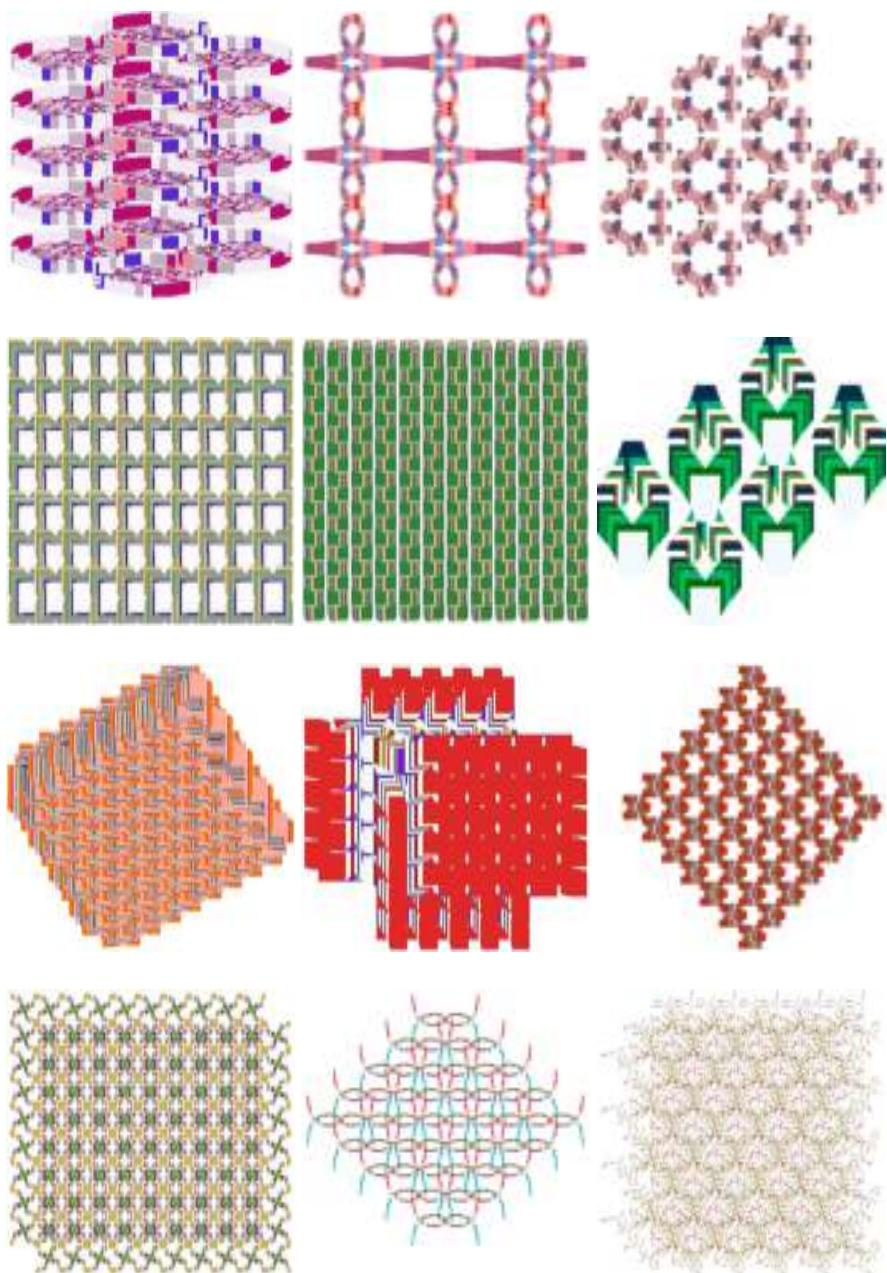


Fig. 9. Sample designs produced with *evoArtlandia* (also see Figure 6 for further designs)

as well as the resulting outcomes—but still remain centered around a human designer. An approach to overcome the evaluation bottleneck in interactive evolutionary computing is discussed in the following section.

6 Sorting Solutions by Spatial Arrangement

Figure 7 shows one of our proposed solutions to create an intuitive and interactive interface for the assessment and categorization of large numbers of design solutions. The evaluation screen is subdivided into fitness zones as illustrated in Figure 7. Each zone is associated with a specific fitness value or a range of fitness values. Solutions are then moved into these respective zones, thus assigning them to a particular fitness value. For example, solutions that end up in zone A will be discarded, whereas solutions in zone F will be given the highest possible fitness values. This way the location of a phenotype window determines its associated fitness. Of course these fitness zones do not need to be rectangular or of equal size, but can take on any shape.

One can even utilize the (x, y) coordinates of the phenotype windows to more accurately specify fitnesses. In Figure 8, all designs that are moved into the bottom right of the screen will be discarded. Four overall fitness categories are defined for the remaining screen area. Within the left half of the screen solutions are ranked according to their vertical coordinates. The higher the top left corner of a phenotype window within a fitness category, the higher is its assigned fitness. This interactive evaluation approach by spatial arrangement can be generalized to a much larger number of design solutions, where it should be easier and much more intuitive for a human ‘breeder’ to categorize these rather quickly (Figure 8). We have used this interface for interactive evaluation to evolve tiling patterns such as in Figure 9 and other artistic designs. This interface has facilitated the use of evolutionary design by artists, engineers, and life scientists, as we have experienced in our collaborations.

7 From an Artist’s Perspective

From an artist’s perspective, evolutionary computing has changed the design process. Traditionally, the designer works toward a specific outcome, incrementally building and adjusting a design step by step as a visual assembler. Now, the designer is a visual sorter, seeking out the best available options from each iteration and massaging the functions to fashion a design trend rather than a specific outcome. Because unimaginable visualizations crop up in the evolutionary process, the computer forces the user to assimilate new design information on the fly. Through iterative permutations, new information not only pushes the look of the design but provides new starting points for redefinition of both concept and outcome. Rapid bombardment by evolving images places your brain on creative-fast-forward. Instead of aesthetic decisions becoming thoughtful and considered, they become intuitive. Because intuition and numerical sorting are polarized ways of processing information, the sorting process for defining the

best visual results moves toward intuitive fuzziness. Whenever a choice can be as simple as drag and drop, it supercedes processes that are less direct and time consuming. The evolutionary iterations are coming up so fast that the designer who stops to type, or to check a box for a pragmatic ranking, breaks the flow. Instead, design happens at a higher level of intuition. The interface that allows the designer to drag and drop into hierarchies that sort information offers the designer a wholeness that does not occur when evaluation is interrupted and segmented.

In the future, we will evaluate the effectiveness and efficiency of the spatial partitioning of the fitness zones. Results described in this paper, along with a gallery of images created with *evoArtlandia* can be found at our website:

<http://www.swarm-design.org>

References

1. Bentley, P., Corne, D.: Creative Evolutionary Systems. Morgan Kaufmann, San Francisco (2002)
2. Khemka, N., Novakowski, S., Hushlak, G., Jacob, C.: Evolutionary design of dynamic swarmscapes. In: Genetic and Evolutionary Computation Conference (2008)
3. Dawkins, R.: The Blind Watchmaker. W.W. Norton and Company, New York (1996)
4. Todd, S., Latham, W.: Evolutionary Art and Computers. Academic Press, Orlando (1994)
5. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of the 18th annual conference on Computer Graphics and Interactive Techniques. ACM Press, New York (1991)
6. Unemi, T.: Sbart 2.4: Breeding 2D cg images and movies and creating a type. Knowledge-Based Intelligent Information Engineering Systems (1999)
7. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. Proceedings of the IEEE 89(9), 1275–1296 (2001)
8. Jacob, C., Hushlak, G., Pilat, J.M.P.M.: Swarmart: Interactive art from swarm intelligence. Leonardo (2007)
9. Kwong, H.: Evolutionary design of implicit surfaces and swarm dynamics. Master's thesis, University of Calgary (2003)
10. Kwong, H., Jacob, C.: Evolutionary exploration of dynamic swarm behaviour. In: Congress on Evolutionary Computation, Canberra, Australia. IEEE Press, Los Alamitos (2003); emergence
11. <http://www.artlandia.com> <http://www.artlandia.com>
12. <http://www.wolfram.com> <http://www.wolfram.com>

Fractal Evolver: Interactive Evolutionary Design of Fractals with Grid Computing

Ryan D. Moniz and Christian Jacob

Computer Science, University of Calgary,
Calgary, Alberta, Canada
{rdmoniz,cjacob}@ucalgary.ca
<http://www.swarm-design.org>

Abstract. Interactive Evolutionary Computing is a powerful methodology that can be incorporated into the creative design process. However, for such a system to be useful, the evolutionary process should be simple to understand and easy to operate. This is especially true in applications where it is difficult to create a mathematical formula or model of the fitness evaluation, or where the quality of the solution is subjective and dependent on aesthetics, such as in the areas of art and music. Our paper explores this idea further by presenting a system that evolves fractal patterns using an interactive evolutionary design process. The result is a tool, *Fractal Evolver*, that employs grid computing and swarm intelligence concepts through particle swarm optimization to evolve fractal designs.

Keywords: Interactive Evolution, Distributed, Parallel, Interface, Particle Swarm Optimization, Fractals.

1 Introduction

This paper demonstrates an application of how an interactive evolutionary process can be utilized to explore the numerous combinations and permutations of fractals [1]. It can be a challenging and daunting task to create, understand, and predict the design of a fractal from a set of parameters, especially considering the large range of parameters and values that are used for fractals in a recursive formula.

It can be extremely difficult to observe a relationship between the parameters of a description of a fractal and what their corresponding effect entails. The fractal in Figure 1(b) results from slight parameter changes from the fractal in Figure 1(a). Comparing the fractals in Figure 1(a) and (c), the extensive modifications to the parameters have yielded major changes to the final rendering of the fractal. What is not apparent, however, is which parameters have the greatest or least effect on potential fractal designs. This is where evolutionary exploration and design tools, as our *Fractal Evolver*, become valuable.

For the very same reasons, it is laborious to manually adjust parameters hoping that one may gain some understanding of how to steer or evolve fractals

(a)		<code>weight="0.5" color="1" symmetry="0" handkerchief="0.247436" aa="0.574084" juliascope="0.178478"</code>	<code>juliascope_power="2" juliascope_dist="1" coeffs="0.18739 -0.931787 -0.0130838 -0.960375 -0.980264 -0.440811 palette_color="73"</code>	<code>weight="0.5" color="0" symmetry="0" fan="0.276528" fan2="0.112714" bubble="0.362664"</code>	<code>cylinder="0.247974" fan2_x="-0.141629" fan2_y="-0.287572" coeffs="-0.501681 -0.733798 0.908606 -0.68777 0.940259 0.489059"</code>
(b)		<code>weight="0.5" color="1" symmetry="0" handkerchief="0.247436" aa="0.1" juliascope="1"</code>	<code>juliascope_power="6" juliascope_dist="3" coeffs="0.18739 -0.931787 -0.0130838 -0.960375 -0.980264 -0.440811 palette_color="234"</code>	<code>weight="0.5" color="0" symmetry="0" fan="0.276628" fan2="0.112714" bubble="0.362664"</code>	<code>cylinder="0.247974" fan2_x="-0.141629" fan2_y="-0.287572" coeffs="-0.501681 -0.733798 0.908606 -0.68777 0.940259 0.489059"</code>
(c)		<code>weight="0.5" color="1" symmetry="0" handkerchief="5.983727" aa="0.574084" juliascope="0.93737"</code>	<code>juliascope_power="16" juliascope_dist="2" coeffs="0.18739 -0.931787 -0.0130838 -0.960375 -0.980264 -0.440811 palette_color="234"</code>	<code>weight="0.5" color="1" symmetry="0" fan="0.82374" fan2="0.93737" bubble="0.74376"</code>	<code>cylinder="0.82136" fan2_x="-0.98421" fan2_y="-0.13244" coeffs="-0.0174.1280 -0.733798 0.908606 -0.68777 0.940259 0.489059"</code>

Fig. 1. Effects of parameter mutation: The original fractal’s parameter settings (a), minor changes to 5 parameter values in (b), and major changes to 14 parameter values in (c)

towards a certain design. In this paper, we present an Interactive Evolutionary Computation (IEC) system that allows for the exploration and optimization of a parametric fractal system. We illustrate how Swarm Intelligence (SI) concepts in the form of particle swarm optimizers can provide assistance in discovering new designs, while also enhancing the design process of fractals.

This paper is organized as follows: Section 2 discusses related works in this field with a focus on fractal flames and Particle Swarm Optimization (PSO). Section 3 explores our work on Grid Computing for efficient fractal generation. Section 4 presents our results thus far and discusses future expansions.

2 Related Work

Fractals can be described as shapes, where small sections of the fractal share a resemblance to the full shape (Figure 2). This concept demonstrates ‘self-similarity’ [1]. Fractals are created using an iterative formula where the output of the formula is redirected as input. Usually interesting patterns emerge from the use of fractal formulas as illustrated in Figure 3 . Fractals have been known for quite some time and have been thoroughly studied by scientists and mathematicians [1], [2].

2.1 Fractal Flames

A new approach to creating fractals –“fractal flames” – was introduced by Draves as part of the “Electric Sheep” project [3]. In fractal flames an iterated function

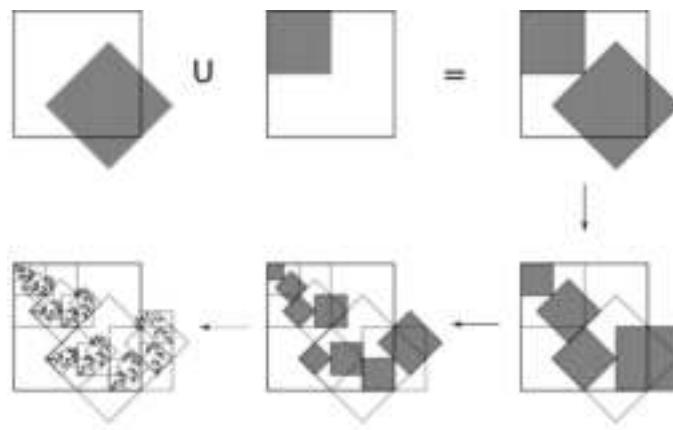


Fig. 2. The fractal starts as a single square but changes after using two functions that transform, scale, and rotate. Three more iterations are performed using the same functions recursively [5].



Fig. 3. The evolution of a fractal flame from generation 0 to 5 using *Fractal Evolver*

system (IFS) is defined for polygons and a series of transformations (rotate, transform, and scale) is iteratively applied. The IFS mapping is defined by $S = \bigcup_i^m f_i(S)$, where $S \subset \mathbb{R}^n$ and $f_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ are the functions to be iterated. S , is the fixed point of a Hutchinson operator, where m functions f_i are applied [4]. The diagram in Figure 2 shows the creation of an IFS from two affine functions. The functions transform the square shape into the shaded square. The merging of the two functions becomes the Hutchinson operator. Some examples of fractal flames created in this manner are illustrated in Figure 3.

The genetic code for a fractal flame consists of 6 transformation functions with 6 affine coefficients, 7 variation coefficients and a weight value. This results in a model of 84 dimensions ($6*(6+7+1)$). In our project we use 103 parameters (Table 1) to evolve fractal flames. Some of these parameters are also shown in Figure 1.

2.2 Interactive Evolutionary Computation

In Interactive Evolutionary Computation (IEC) the mathematical formula model of the fitness evaluation is replaced or augmented by a human evaluator. Dawkins

Table 1. The complete set of default parameters for creating fractal flames and their range values. Fields denoted by an asterisk (*) are created m number of times depending on the number of xForms specified.

Parameter	Min	Max	Parameter	Min	Max
Scale	0	100000	Co-efficient 1X*	-100000	100000
Time	0	10000	Co-efficient 1Y*	-100000	100000
Fractal Symmetry	0	2	Co-efficient 2X*	-100000	100000
Brightness	0	10000	Co-efficient 2Y*	-100000	100000
Gamma	0	100000	Co-efficient 3X*	-100000	100000
Gamma Threshold	0	1	Co-efficient 3Y*	-100000	100000
Hue	0	1	Post Co-efficient 1X*	-100000	100000
Palette Selection	0	699	Post Co-efficient 1Y*	-100000	100000
Vibrancy	0	100000	Post Co-efficient 2X*	-100000	100000
Batches	0	59	Post Co-efficient 2Y*	-100000	100000
Estimator	0	5	Post Co-efficient 3X*	-100000	100000
Estimator Curve	0	1	Post Co-efficient 3Y*	-100000	100000
Estimator Minimum	0	1	xForms Symmetry*	0	100000
Filter	-1	1	Weight*	0	1
Filter Type	0	13	Co-efficient*	0	1
Motion Exponent	0	59	Final XForm*	0	1
Oversample	1	59	Variation Type	0	38
Temporal Samples	0	59	Number of xForms	1	6

was one of the first to apply IEC to artistic design [6]. Naturally, this approach led to other visual IEC systems like Sims' genetic art [7], or Latham's Mutator [8]. A similar system to evolve fractals was made by Lutton called ArtiE-Fract [9] using an Evolutionary Algorithm and Evolutionary Strategies [10] to evolve fractals.

At the time of Dawkins, Sims, and Latham computing resources were limited to a single computer, and the interface for evolution was constrained to selecting one individual to mutate. Our system utilizes the population dynamics of multiple potential solutions. Although one solution will be ranked the highest out of the entire population, the solutions that are ranked lower will still contribute to the evolution of the next generation of fractal designs. This is due to our choice to use Particle Swarm Optimization as our population-based search algorithm.

2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was inspired by the manner in which bird flocks move in synchronization and maintain their distance from neighboring birds [11]. In the PSO algorithm, each of the “birds” fly through an n -dimensional search space. To understand the PSO search dynamics, we illustrate the location update for the particles, each of which represents a solution in n -dimensional space. The particle location, X_n , and velocity vector, V_n , are updated for each subsequent timestep as follows:

$$X_n(t + \Delta t) = X_n(t) + V_n(t) * \Delta t \quad \text{where } n = 1, \dots, N. \quad (1)$$

The velocity vector for each of the particles is updated by:

$$V_n(t + \Delta t) = V_n(t) + c_1 * \text{Random}[0, 1] * (G^{(best)}(t) - X_n(t)) + \\ c_2 * \text{Random}[0, 1] * (P_n^{(best)}(t) - X_n(t)). \quad (2)$$

The Δt parameter determines the discrete time interval for a particle's movement, where higher granularity results in a larger area being searched in fewer time steps. Two attractive forces determine the direction of each particle: the global best solution, $G^{(best)}$, and the personal best solution, $P_n^{(best)}$. This process is performed for each element of the velocity vector. The uniformly distributed random numbers, $\text{Random}[0,1]$, help the particle move through the solution space by randomly giving more or less emphasis to the global or personal best solutions, defined by the weight constants c_1 and c_2 , respectively. The key here is that the particles swarm around the solution space, gravitating towards the global and personal best solutions. The stochastic elements permit the particles to cover more of the solution space, while maintaining proximity to the best known solutions. The PSO algorithm is very efficient, computationally inexpensive, and excels at tuning parameter type problems better than some traditional evolutionary algorithms [12].

3 Grid Computing of Fractals

PSO has been shown to be effective for solving complex parameter optimization problems, usually comprising of a large number of parameters [13]. Typically, when performing a lot of optimization runs, additional computation power is required. Rendering fractals in sufficient detail and gradable quality is an expensive computation. Calculating fractals can require extensive computing resources. A 640×480 pixel image contains over 300,000 points, where each of these points is repeatedly recalculated through the IFS over several hundred times. As a result, an image would require more than 300,000,000 calculations. Depending on the complexity of the parameter settings and the processing power of the computer used. Therefore, rendering times for fractals can vary from several minutes to several hours. As a consequence we need a method to speed up calculations. This can be remedied through the use of grid computing.

Grid computing utilizes distributed computing resources through groups of independent computers that appear to form a large virtual supercomputer. The computers on the grid can be geographically dispersed around the world, thus providing enormous flexibility. Rendering images such as fractals is ideal as it is an independent task that does not require communications with other machines for completion.

Our project utilizes Xgrid, a grid computing technology built into Mac OS X that facilitates the creation and organization of a grid of computers to run computationally intensive applications, such as the rendering of a large number of fractals [14]. Xgrid leverages a three-tiered architecture approach to grid computing as seen in Figure 4.

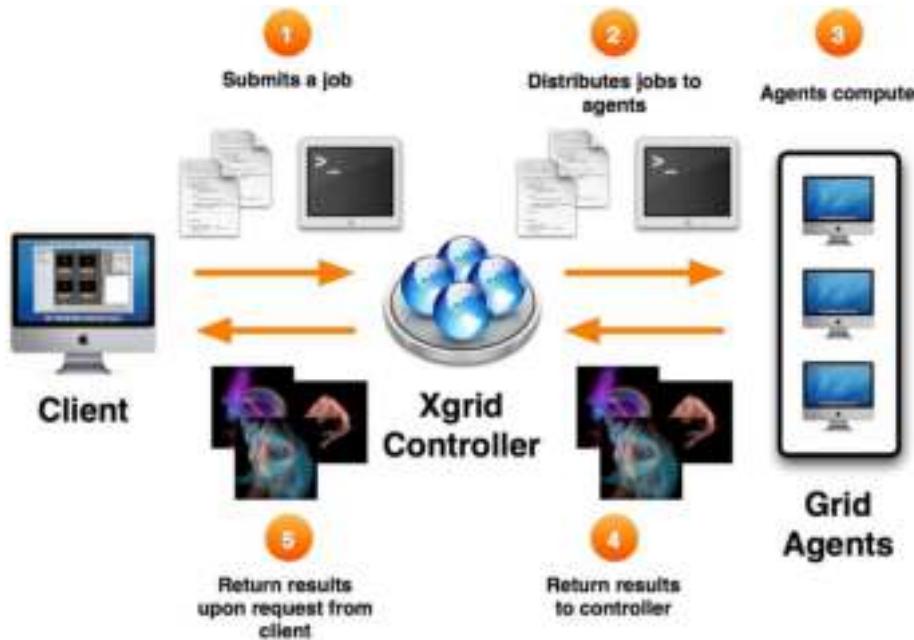


Fig. 4. *Fractal Evolver's* process overview involving Grid Computing

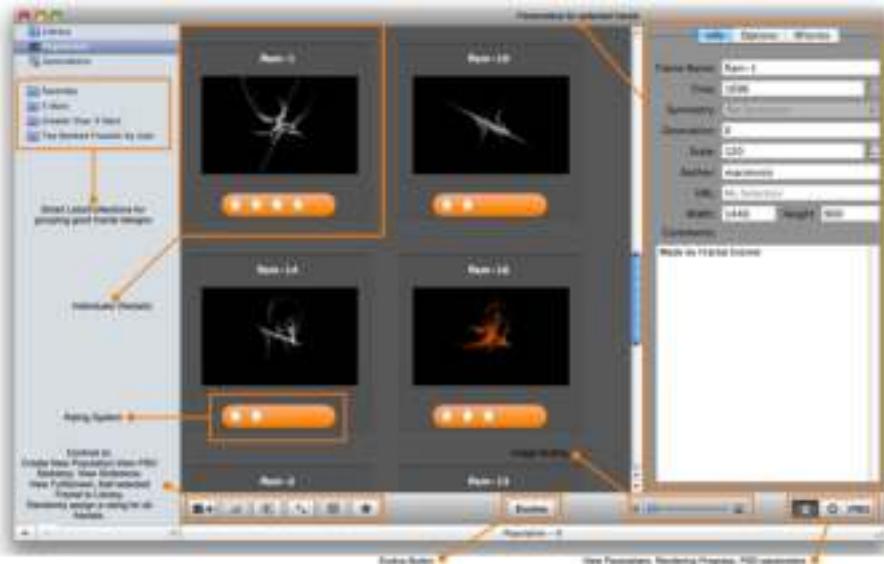


Fig. 5. *Fractal Evolver* interface: The designer can view fractals in a grid like fashion, view parameters, and assign a ranking score of 1 to 5 stars to each fractal

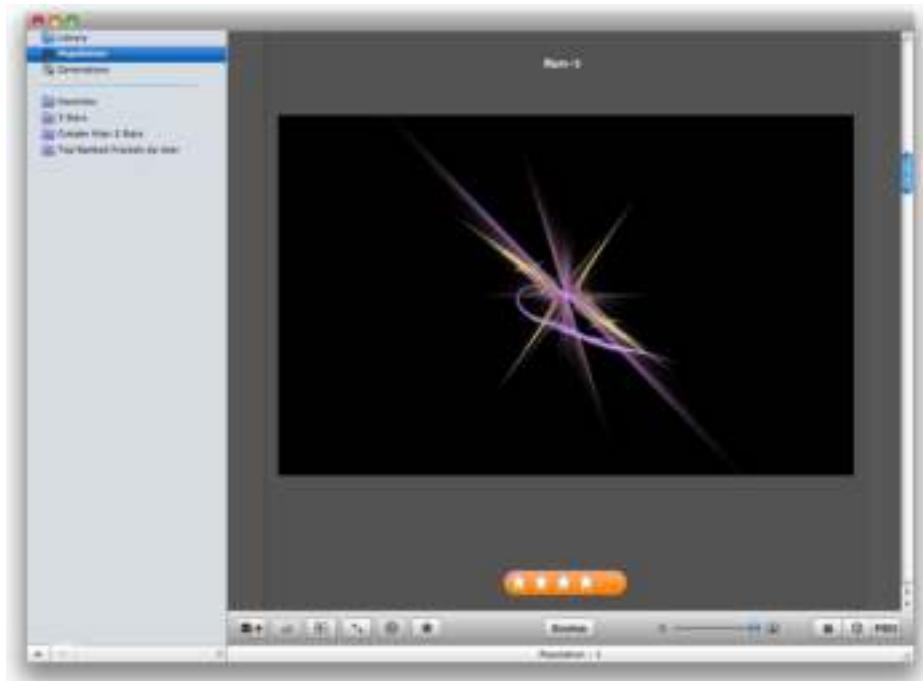


Fig. 6. *Fractal Evolver* interface: Viewing a fractal in full resolution and assigning a ranking score of 4 stars

Fractal Evolver generates fractal designs and relies on an evaluator to assess the fitness value for each fractal in the population (Figure 5). The initial process begins by specifying how many individuals (fractal flames) to use for the population. The fractal genomes are then created and submitted to the Grid for rendering (Step 1 in Figure 4). The Xgrid Controller receives all the fractal genome files and begins to distribute the task to all available Agent computers in the Grid for rendering (Step 2). As each fractal is rendered and completed (Step 3), the Agent sends the image to the Xgrid Controller, which in turn is sent to the Client for evaluation (Steps 4 & 5).

In the graphical control interface, fractals are displayed in a grid-like fashion (Figure 5). Each of the fractals can be resized to full image resolution for more detailed evaluation (Figure 6). The designer or “breeder” can assign a fitness evaluation of 1 to 5 stars to each fractal, 1 being the lowest and 5 being the highest fitness evaluation. After at least one fractal has been ranked, the designer will invoke the “Evolve” button, and the interactive PSO algorithm will iterate through all the individuals, evolving the parameters following the PSO algorithm as described before. This new generation of fractals will then be submitted to the Grid to be rendered. This process is repeated until the designer is satisfied with the resulting fractals.

4 Results and Conclusion

In our tests the use of Grid Computing has significantly accelerated the IEC process by generating a diversity of results, with much reduced response times. This is particularly important with a human in the evaluation loop (Figure 7). Our evolutionary process saw a speedup of at least 3.5 when working with large population sizes of up to 640 fractals. In comparison, the serial processing approach delays the evolutionary process by rendering each fractal sequentially forcing the user to wait until the last fractal is rendered for assessment. As seen in Figure 7, generating 640 simple fractals with image resolutions of 1440×900 sequentially takes more than 2 hours to complete. Using the distributed approach, the rendering takes about 30 minutes. Typically, with any interactive system, the designer may suffer from fatigue having to observe and evaluate all the possible designs that the evolutionary system produces [15]. Reducing the system's response time has helped to alleviate the fatigue that designers experience, while one is able to go well beyond simple fractal designs that might render more quickly, but do not produce interesting results. The easy setup and use of grid computing to distributively render fractals in parallel speeds up the evolutionary process as the

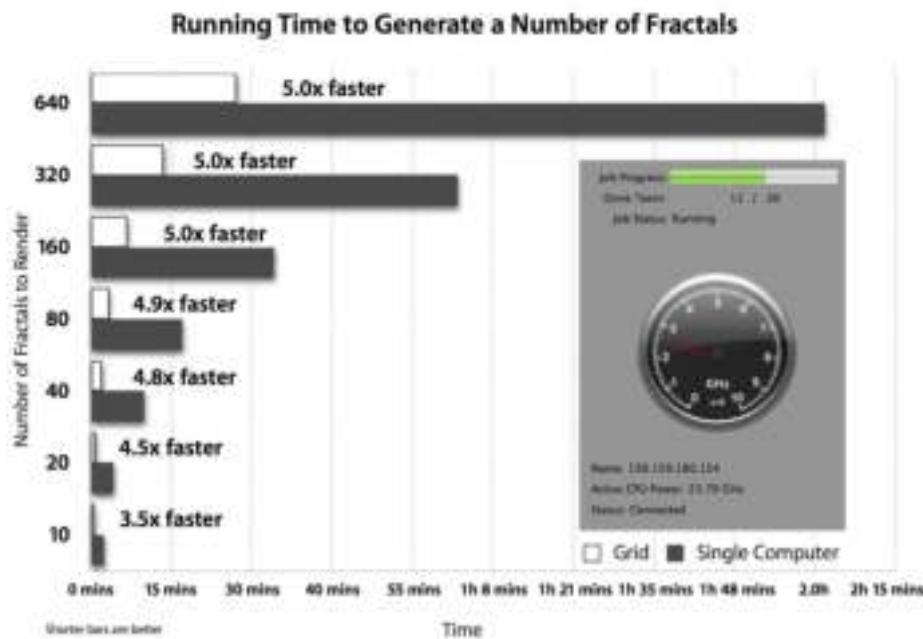


Fig. 7. Experiments were conducted on a single computer (Duo Core 2GHz & 2GB RAM) and on a grid of computers (230GHz & 113GB RAM). Grid computing resulted in less time spent waiting for fractals with resolutions of 1440×900 to be generated compared to a single computer, on average 4.7x faster. Composed onto the graph is the progress view from *Fractal Evolver* that displays the current job progress when fractals are distributed to the grid for rendering.

user does not have to wait for each of the individual fractals to be rendered one after the other. The distributive approach also increases the number of potential solutions generated, which gives the user more options to find a particular design that they will like.

In terms of evolution, the PSO algorithm has been proven effective in dealing with fine-tuning a large parametric system such as the fractal flames. The intuitive interface with its simplified grid view browser for observing fractal designs easily allows users to examine and assess a large population of fractals (Figure 5). The browser is particularly useful after the fact, when one needs to inspect all the fractals produced by the system. The all-in-one design provides parameter information for on demand analysis of the current evolutionary process without cluttering up the computer screen with multiple windows. Assigning a score of 1 to 5 stars to the population gives the PSO algorithm more flexibility to explore the design space. Ideally, the user will pick one fractal image from the population and assess it a score of 5 stars indicating that this is the best fractal design. The 5 star ranking system was selected as our method to classify and evaluate the quality of fractal designs because 5-point scales are commonly used by websites, applications, restaurants, hotels, movies, etc. [16]. The 5 star ranking system provides a linear visual form and data that has been ranked using the 5 star method are easier to sort and compare.

These experiments clearly show the need to balance the advantages of human evaluation with an evolutionary system. Our further research will attempt to minimize the amount of human interaction with the evolutionary system by utilizing both a machine learning component, as well as having the designer only evaluating every n -th generation. The project was built with the intention of being able to easily replace the fractal design problem by any other compute-intensive task. Recent developments have shifted to also include automatic evaluation for projects that do not require aesthetic appraisal. The same IEC system will be applied to evolving a parametric gene regulatory system that simulates biomolecular interactions within an *E.coli* bacterial cell [17]. We will then take what we have learned from evolving the gene regulatory system and introduce an automatic evaluation component to pre-filter undesirable fractals. This will reduce the number of fractals that the user will have to assess overall because only the fractal designs that have passed the user's automatic evaluation criteria will be displayed.

Acknowledgments. We would like to extend our thanks and appreciation to Namrata Khemka and Navneet Bhalla for their continued support and encouragement.

References

1. Peitgen, H.O., Richter, P.H.: *The Beauty of Fractals. Images of Complex Dynamical Systems*. Springer, Heidelberg (1986)
2. Flake, G.W.: *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MIT Press, Cambridge (2000)

3. Draves, S.: The electric sheep and their dreams in high fidelity, Annecy, France. ACM Press, New York (2006)
4. Peitgen, H.-O., Jürgens, H., Saupe, D.: Chaos and Fractals. Springer, Heidelberg (2004)
5. Draves, S.: Electric sheep, <http://www.electricsheep.org>
6. Dawkins, R.: The Blind Watchmaker. Penguin, London (1986)
7. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of SIGGRAPH 1991. ACM Press, New York (1991)
8. Todd, S., Latham, W.: Evolutionary Art and Computers. Academic Press, London (1992)
9. Lutton, E., Cayla, E., Chapuis, J.: Artie-fract: The artist's viewpoint. In: Evo Workshops [12], pp. 510–521
10. Engelbrecht, A.: Computational Intelligence: An Introduction. Halsted Press, New York (2002)
11. Eberhart, R.C., Kennedy, J.: A new optimizer using particles swarm theory. In: Proc. Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43. IEEE Service Center, Piscataway (1995)
12. An Evolutionary Modularized Data Mining Mechanism for Financial Distress Forecasts. Studies in Fuzziness and Soft Computing, vol. 163. Springer, Heidelberg (2006)
13. Khemka, N.: Comparing particle swarms and evolution strategies: Benchmarks and application, Master's thesis, University of Calgary, Calgary, AB, Canada (2005)
14. Kramer, D., MacInnis, M.: Utilization of a local grid of mac os x-based computers using xgrid, hpdc, vol. 00, pp. 264–265 (2004)
15. Kamalian, R., Zhang, Y., Takagi, H., Agogino, A.M.: Reduced human fatigue interactive evolutionary computation for micromachine design. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 9, pp. 5666–5671 (August 2005)
16. Pu, P., Punit, G.: Visualizing reputation and recommendation in scientific literature. In: 10th International Conference on Human - Computer Interaction, Crete, Greece (2003)
17. Jacob, C., Burleigh, I.: Biomolecular swarms - an agent-based model of the lactose operon. Natural Computing 3(4), 361–376 (2004)

Humorized Computational Intelligence towards User-Adapted Systems with a Sense of Humor

Pawel Dybala, Michal Ptaszynski, Rafal Rzepka, and Kenji Araki

Graduate School of Information Science and Technology, Hokkaido University

Kita 14 Nishi 9, Kita-ku, 060-0814 Sapporo, Japan

{paweldybala, ptaszynski, kabura,
araki}@media.eng.hokudai.ac.jp

Abstract. This paper investigates the role of humor in non-task oriented (topic restriction free) human-computer dialogue, as well as the correlation between humor and emotions elicited by it in users. A joke-telling conversational system, constructed for the needs of this research, was evaluated by the users as better and more human-like than a baseline system without humor. Automatic emotive evaluation with the usage of an emotiveness analysis system showed that the system with humor elicited more emotions than the other one, and most of them (almost 80%) were positive. This shows that the presence of humor makes computers easier to familiarize with and simply makes users feel better. Therefore, humor should be taken into consideration in research on user-friendly applications, as it enhances the interaction between user and system. The results are discussed and our concept of a user-adapted humor-equipped system is presented.

Keywords: Humor, Human-Computer Interaction, emotions, conversational systems, computational humor.

1 What Makes Humans Humans?

To dispel the reader's fears at the very beginning – this section is not aimed at answering the age-long question about the nature of human beings. What we are going to do below, however, is to discuss some features of interaction between humans and introduce our research in which we examined their role in interactions with machines. The results described in this paper show that we took one step further towards Humanized Computational Intelligence.

1.1 Chatting as a Part of Interaction

We, humans, have developed an ability to communicate with each other through the usage of language. Most of us do that every day, with different purposes. Some of our conversations are clearly task-oriented, while others just concentrate on maintaining social relations, amusement and pleasure coming from interaction itself. Having no talking partner is commonly associated with loneliness, sadness, and generally bad emotions, which can be proved easily by putting the phrase “no one to talk to” into

Google (or any other Internet search engine). It is not a coincidence that most psychological therapies are based on conversation. However, it is the opportunity to talk just for fun (to chat), without a particular purpose, that actually makes us feel that we are part of society.

1.2 Humor as a Positive Factor

We are far from promoting humor as a cure for everything – there are, however, many robust proofs for its beneficial influence on our life. To name only few of them, we know that humor is often used as a measure to cope with negative emotions and moods, such as stress [1], anxiety, or depression [2]. Research conducted by Vilaythong et al. showed that exposing people to humorous contents (like funny videos) increased their feeling of hopefulness and obviously made them feel better [3]. Proofs for social benefits of humor were also provided by Cook and Rice [4] – the results of their experiments showed that a sense of humor in another person increases the perceived benefits of a relationship. According to Sprecher and Regan [5], the sense of humor is also one of the main characteristics we use when choosing a partner, which means that we like to interact with people who use humor. Finally, we have the research of Mulkay [6], who proved that we tend to use jokes when discussing difficult subject with others, which leads to the conclusion that humor actually makes our conversation easier.

2 What Makes Computers More Human?

The issues of human-likeness described in section 1 have not been mentioned here without reason – both the ability to chat and to use humor are said to be applicable such areas as Human-Computer Interaction or Humanized Computational Intelligence.

It has been convincingly demonstrated that humans treat computers as if they were social actors. According to SRCT (Social Response to Communication Technologies) theory, people respond to computers using the same social attitudes and behaviors they apply to other people [7]. This leads us to the statement that – if computers are really treated (or going to be treated) in similar way to humans – the role of mere tools in our lives is not the only one to be played by them. If we approach them as to social actors, it is obvious that we expect them to behave in as natural a way as possible, and the interaction between us should then go smoothly. Therefore, if we chat with other people, we would also be willing to do it when communicating with computers, and if we frequently exploit the benefits of humor in the interaction with other humans, they should have the same effect in interaction with machines.

“Machine talking” itself is a subject of research in the field of NLP (Natural Language Processing) – however, even here non-task oriented (topic or task restriction-free) discourse is still an underestimated issue. Implementing humor into such non-task oriented systems is an even more neglected area, and we believe it is a subject worthy of more intensive research.

3 Pun-Telling Freely Talking System

In our research we decided to fill that gap. We started a project aiming to construct a funny, humor-equipped freely talking system. For the subject of this enterprise we chose puns – a genre of jokes which is based on features of the language (such as homophony) and thus is relatively computable with the current methods of NLP. The base language of our research is Japanese, as it is rich in homophonic phrases which gives vast possibilities of generating puns (in Japanese called *dajare*).

Having constructed a simple joke-telling conversational system (described in our previous works [8] - see section 4.2 for a description of the algorithm), we checked its performance in an evaluation experiment, in which users talked first with the system without humor, and then with a similar one with humor. Next, we analyzed the chat logs with our emotiveness analysis system to check if the presence of humor actually influenced the user's emotions. Our premise was that, if successful, the users' emotions towards the system with humor should be more positive than in the case of the non-humorous one. The results are described below.

4 Systems Used in This Research

In this research we used two talking systems: with and without humor. The latter is equipped with our pun generating engine (see section 4.2). To analyze the results, we used an emotiveness analysis system, which recognizes emotions in utterances (see section 4.4).

4.1 Freely Talking System

The baseline system in this research is a text-based chatterbot called “Modalin” (developed by Higuchi et al. [8]) which uses the Internet to extract word associations for interlocutor utterances and adds modality to generated responses. For example, given an user input phrase: “- *Nanika sukina tabemono aru?* (What food do you like?)”, the system extracts a keyword: “*tabemono*” (food), checks its associations on the Internet, finds the word “*oishii*” (tasting good) as the closest one to the keyword, and uses it to generate a sentence. Then, it adds modality to the generated phrase, and outputs the response, which in this case is: - *Maa, tabemono-wa oishii desu.* (Well, food tastes good.).

In the evaluation experiment, human subjects assessed over 80% of the extracted associations as correct. The experiment also showed that adding modality to the system’s response improved its results in all criteria.

4.2 Pun Generator

The pun generating engine PUNDA, developed in our previous research [9], is also based on the Internet. From a user utterance it extracts a base word and transforms it using Japanese pun phonetic generation patterns, to create a phonetic candidate list.

Then, it checks all candidates in the Goo search engine¹, and chooses the one with the highest hit rate, i.e. the most common word that sounds similar. Next, it uses the KWIC on WEB - online Keyword-in-context sentences database [10] – to find a sentence with the chosen word and extracts part of the sentence starting with the word.

Below we present an example of the system in action:

User: *Jaa, issho ni eiga wo mi ni ikanai?*
(So, will you go to see a movie with me?)
 Base word: *eiga* (a movie)
 Pun candidate: *eiga* (glory)
 KWIC Sentence part: ...*eiga wo hokotta* (was glorious)
System's response: *Eiga (movie) to ieba eiga (glory) wo hokotta.*
(Speaking of movies, it was really moving!)

If no candidate was found using the user's input, the system randomly selects a pun from our pun data base.

4.3 Freely Talking Joking System

The two algorithms described above were combined to create a talking system which tells jokes. The system was named "Pundalin". As far as joke timing is concerned, we decided to apply a very simple rule – in every third turn of the conversation, the normal talking system's output was replaced by a joke, generated by the joking system. In other words, the user's every third utterance becomes an input for the joke generator, which generates an appropriate pun for it. Preliminary tests showed that joking in every third turn is optimal for this experiment. The system has also been described in our previous work [9].

4.4 Emotiveness Analysis System

To check what emotions our humor-equipped talking system triggered in the users, we used the Emotive Elements/Emotive Expressions Analysis System (ML-Ask), which determines emotiveness of utterances. The system was developed as a part of a previous project [11]. Based on some ideas presented in our earlier publications, the system performs utterance analysis in two general steps: **1. Determining its general emotiveness** (emotive/non-emotive) and **2. Specifying the types of emotions found** (in emotive utterances only).

In the **first step**, if many of the user's utterances were determined as emotive, it was assumed that he or she was emotionally involved in the dialogue. In Japanese, emotional engagement in the conversation suggests a tendency to familiarize with the partner [12] – which, in this case, is the talking system.

In the **second step**, analysis of the specific emotions showed by the evaluators during the conversation provides us with the information of their feelings towards the interlocutor (in this case – the talking system). If the detected emotions were positive or changing from negative through neutral (non-emotive) to positive during the whole conversation, the general sentiment towards the talking system was considered as

¹ www.goo.ne.jp

positive. If detected emotions are negative or changing from positive through neutral to negative during the whole conversation, the general sentiment towards the talking system was considered as negative.

5 Checking If It Works

We asked 13 people (university students) to perform a conversation with Modalin (without humor, referred to as System A) and with Pundalin (with humor, referred to as System B). As the dialogue was supposed to be as free as possible, no topic restrictions were made. Next, users filled out a questionnaire about the dialogue. Then, the chat logs were checked with the emotiveness analysis system for the presence of different types of emotions.

In previous experiments we asked third person (non-user) human evaluators to tag the emotions in the utterances manually – this, however, is quite laborious and often very subjective, as people tend to annotate emotions with a personal attitude. Using the emotiveness analysis system in the evaluation not only lets us reduce human labor, but also allows us to get more robust and objective results.

6 Did It Work?

6.1 User's Point of View

A summary of the user's questionnaire was given in our previous work. Among the questions asked in this part of the evaluation, 5 are of high relevance to the subject discussed in this paper²: 1) Do you want to continue the dialogue? 2) Do you think that the system was human-like? 3) Do you think the system tried to make the dialogue more funny and interesting? 4) Did you find the system's talk interesting and funny? and 5) Which system do you think was better? The first question concerned the willingness to interact with the computer, which obviously is desirable in the field of designing user friendly systems – we can construct extremely sophisticated engines, but it will mean literally nothing if no one would like to interact with them. In the second question we asked directly about the human-likeness of the system. Of course, there is no one and firm definition of this category, and we do not actually aim to find any, as we decided to focus on the user's subjective impression here. The same approach was employed in the following questions, which, albeit quite general, simply give us the answer – which system does the user choose?

The answers to the first four questions were given in a 5-point scale, while for the last one the user only had to choose one of the two systems. Results are shown in Table 1. Statistic significance of all results was checked using the Student's t-test. All differences (apart from question 1. - p value=0.06) were found to be statistically significant on 5% level, which means that the results are robust.

The scores clearly show that users appreciated the humor-equipped system higher than the one without humor.

² Other questions concerned the system's linguistic quality, such as grammatical correctness, and are not relevant to the subject of this paper.

Table 1. User's answers for questions: 1) Do you want to continue the dialogue? 2) Do you think that the system was human-like? 3) Do you think the system tried to make the dialogue more funny and interesting? 4) Did you find the system's talk interesting and funny? and 5) Which system do you think was better? Answers were given on a 5-point scale

Questions	1	2	3	4	5
System A	2.62	2.38	1.92	2.46	2 users (15%)
System B	3.38	3.31	4.15	4.08	11 users (85%)
difference	0.76	0.93	2.23	1.62	

6.2 The Emotive Effect

The tendencies shown by the results of emotive analysis are similar to those gained in the human evaluation. Almost all evaluators were more emotionally engaged in the conversations with System B (Pundalin) - see Figure 1.

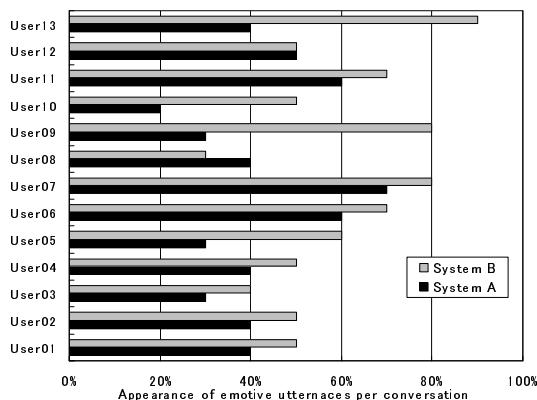


Fig. 1. Average percentage of appearance of emotively engaged utterances for all human evaluators in conversations with System A (without humor) and System B (with humor)

Also the analysis of specified types of emotions clearly showed that evaluators' attitudes to System B were generally positive (78%), while most of the attitudes toward System A (Modalin) were negative (75%) - see Figure 2. The results are discussed in section 7.



Fig. 2. The total relation of positive emotions to negative conveyed in the utterances of human evaluators with System A (non-humor-equipped) and System B (humor-equipped)

7 What Does It Actually Mean?

The results described above clearly show the supremacy of the humor-equipped system. Of course, the user's point of view should be seen as the most important criterion – it is the user who is going to interact with the system after all. In such understanding, the results of the user experiment are highly satisfying. Increases in human-likeness, willingness to continue the dialogue and overall assessment of the system prove that we have made a step in the right direction.

The results of the user experiment have been discussed in one of our previous works [9]. Here we would like to focus on the emotiveness analysis results and their implications for the field of Humanized Computational Intelligence.

7.1 Affect-as-Information

To understand the results of the emotiveness analysis experiment, we have to take a look at the so-called “affect-as-information” approach, proposed by Schwarz and Clore [13]. The main idea of this approach is based on a claim that humans use affect in the same way as any other criterion, namely – by using the informational value of their affective reactions to form their opinions and judgments. This leads to the assumption that information about someone's attitude to a product can be derived from information about changes in his or her affective states during its usage [14]. Subsequently, according to Sugiyama [12], if we show more emotions towards someone or something, this means that we tend to familiarize with the object.

This leads us to the conclusion that, since the system with humor elicited much more emotions in the users than the one without humor, it can be stated that it is easier to familiarize with it as an interlocutor - which, in fact, is consistent with the claims presented by Sprecher and Regan [5] (mentioned in section 1.2).

7.2 System That Makes Users Feel Better

Thus, we have a humor-equipped talking system that induces more emotions and therefore is more likeable than the other one. This, however, could lead us into a trap: what if most emotions elicited by the system were negative? Obviously, we would acquire a more human-like machine, but not the kind we would like to interact with. This is why we analyzed also the specific types of emotions. The results showed that most emotions induced by the system with humor were positive – and this is what assures us that the effects of our experiment are in fact desirable. In other words, we managed to create a talking system which makes the user feel better. This proves that the beneficial features of humor actually work well also in interaction between humans and machines and should not be neglected in research on user-friendly applications.

8 Emotive-Analysis-Based Evolution of Humor: Adapting to the User's Sense of Humor

Although the experiment described above was a success, obviously many improvements can still be made. The next stage of our research is aimed at creating a

more human-like and more natural joking conversational system that would be able to learn what puns match the preference of the particular user, to recognize when to tell a pun, and to make the user feel better with humor in a more “clever” way than it does now. In other words, the system should be able to adapt to the user. Below we present the outline of our new approach (see sections 8.1, 8.2 and Figure 3).

8.1 Evolving Individualized Sense of Humor

In the experiment described in the previous sections we used an emotiveness analysis system to determine the user's emotional states. However, the applicability of this system is not limited to the evaluation of dialogues – the results of analysis can also be used in a sense of humor evolution algorithm. During the dialogue, the system can analyze each user's utterance, and check how he/she reacts to particular types of jokes. On this basis, the system can build a representation of the user's sense of humor, determining the types of jokes the user likes/dislikes. For example, if the user reacts with positive emotions to jokes concerning politics, the system can assume that this type of joke matches his/her sense of humor. In this manner, the longer the system would talk to the user, the more accurate “tags” of humor sense it can attach – and this, in effect, shall lead to more personalized, more individualized jokes that with a high probability would be appreciated by the user.

8.2 Reacting to User's Emotions

In its current shape, our joking system uses humor at every third turn of the conversation, which obviously has to be changed. Also here we are planning to make use of the emotiveness analysis system, in order to determine the appropriate timing of jokes. We are now conducting experiments to determine which emotional states allow using humor. From previous research we know that humor can be employed to cope with such negative emotions as stress or anxiety. Also the experiments described in this paper showed that humor can alter the emotions from negative to positive. This leads us to the assumption that the emotive analysis system can be used to determine if the user's current emotional state allows for telling him/her a joke.



Fig. 3. Outline of the emotive-analysis-based evolution concept (for the humor-equipped conversational system Pundalin)

This idea can be combined with the concept of evolving an individualized humor sense (see section 8.1). Another “tag” in the user’s humor sense model can be based on the types of emotions which in the case of this particular user respond well to humor. For example, if the user is angry and does not tend to calm down under the influence of humor, it can be stated that this particular type of emotion does not allow for telling jokes in this user’s case. In this manner the system will be able to adjust to the users’ needs, which should be highly appreciated by them.

Of course, the amount of tag types is not limited and does not necessarily have to be related to humor. The possibilities given by using the emotiveness analysis system are vast and we believe that this topic should be a subject of thorough discussion.

9 Possible Applications

Final goals of research projects focused on freely talking machines are often misunderstood. Creating a talking-only-system that a user would launch in order to perform a few turns of conversation, is not what we are planning to achieve here. Such systems are only a step towards a more important aim, which is equipping computers (and generally machines we interact with) with a possibility to talk with us in a free (non-task-oriented) way, and, in case of this particular research, to use humor. If we are to treat computers as social actors, they should be able to chat with us and also to use and understand humor, but it does not mean that they should do only that, in a persistent and stubborn way. Therefore, the system presented in this paper should be seen rather as a vessel, created to test some mechanisms and possibilities of creating a functional talking machine. In this meaning, the results we acquired are of high importance and make a good starting point for further research.

As mentioned above, if we are to give chatting and joking possibilities (not obligations!) generally to machines that interact with us, possible applications of such systems are limitless. However, there are some particular cases in which the functions of chatting and using humor are of high importance. There have been some inquiries from car-making companies, interested in creating an intelligent, user-friendly car navigator, able to perform the function of being the driver’s conversation partner. Such devices (currently under development) will be able to detect when the driver is getting sleepy, and to do something to wake him/her up. One way of activating driver’s brain and preventing him/her from losing attention would be the use of humor. Therefore, implementing a humor-equipped talking system into a car navigator seems like a very promising idea and, in fact, it has already become the subject of car companies’ interests.

10 Conclusion

In this paper we investigated the potential of humor in human-computer interaction. The results of our research are consistent with existing literature on this subject and showed that the communication between humans and machines can be significantly enhanced with humor. Automatic emotiveness evaluation showed that after having talked with the joking system users conveyed more positive emotions than in the case

of the non-humorous one. The next phase of our research is aimed at creating a user-adapting system with an individualized humor sense (presented in section 8).

In our research we focus on puns as the most easily computable genre of jokes; however, we believe that some mechanisms developed in this project are applicable also to other types of humor. The results presented in this paper, for example, confirmed one very important hypothesis: even very simple (pun-based) humor can visibly enhance interaction between humans and computers – which obviously should work also for other types of jokes and humor in general. This gives our research more universal dimension and is reason enough to proceed further with this project.

References

1. Cann, A., Holt, K., Calhoun, L.G.: The roles of humor and sense of humor in responses to stressors. *Humor* 12(2), 177–193 (1999)
2. Moran, C.C.: Short-term mood change, perceived funniness, and the effect of humor stimuli. *Behav. Med.* 22(1), 32–38 (1996)
3. Vilathong, A.P., Arnau, R.C., Rosen, D.H., Mascaro, N.: Humor and hope: Can humor increase hope? *Humor* 16(1), 79–89 (2003)
4. Cook, K.S., Rice, E.: Social exchange theory. In: Delamater, J. (ed.) *Handbook of social psychology*, pp. 53–76. Plenum, New York (2003)
5. Sprecher, S., Regan, P.C.: Liking some things (in some people) more than others: Partner preferences in romantic relationships and friendships. *J. Soc. Pers. Relat.* 19(4), 463–481 (2002)
6. Mulkay, M.: On humor: Its nature and its place in modern society. Basil Blackwell, New York (1988)
7. Reeves, B., Nass, C.: *The media equation: How people treat computers, television, and new media like real people and places*. Cambridge Univ. Press, Cambridge (1996)
8. Higuchi, S., Rzepka, R., Araki, K.: A Casual Conversation System Using Modality and Word Associations Retrieved from the Web. In: *Proceedings of the EMNLP 2008*, Honolulu, USA, pp. 382–390 (2008)
9. Dybala, P., Ptaszynski, M., Higuchi, S., Rzepka, R., Araki, K.: Humor Prevails! - Implementing a Joke Generator into a Conversational System. In: Wobcke, W., Zhang, M. (eds.) *AI 2008. LNCS (LNAI)*, vol. 5360, pp. 214–225. Springer, Heidelberg (2008)
10. Yoshihira, K., Takeda, T., Sekine, S.: KWIC system for Web Documents (in Japanese). In: 10th Annual Meetings of the Japanese Association for NLP, pp. 137–139 (2004)
11. Ptaszynski, M., Dybala, P., Shi, W., Rzepka, R., Araki, K.: Disentangling emotions from the Web. Internet in the service of affect analysis. In: *KEAS 2008*, Nagaoka, Japan, pp. 51–56 (2008)
12. Sugiyama Lebra, T.: *The Japanese Self in Cultural Logic*. Univ. of Hawaii Press (2004)
13. Schwarz, N., Clore, G.L.: Mood, misattribution, and judgments of well-being: Informative and directive functions of affective states. *J. Pers. Soc. Psychol.* 45, 513–523 (1983)
14. Jiao, J., Xu, Q., Du, J.: Affective Human Factors Design with Ambient Intelligence. In: *HAAI 2007*, pp. 45–58 (2007)

Innovative Chance Discovery – Extracting Customers' Innovative Concept

Hsiao-Fang Yang and Mu-Hua Lin

Department of MIS, National Chengchi University, NO.64, Sec.2, ZhiNan Rd., Wenshan District, Taipei 11605, Taiwan
{94356507, 95356503}@nccu.edu.tw

Abstract. The brainstorming is a useful method to collect customers' ideas, and the interactive evolutionary computing (IEC) usually evolves into the personal innovative product according to the designer's preference. In this study, we follow the grounded theory to formulate the framework of interactive chance discovery (ICD). After collected the evolution data, we use social network analysis (SNA) indexes to identify strong-tie and weak-tie's relationship. According to the phenomenon of the small world, we believed these complicated relationships include some images of present and future. And these images can help us to discover the market of future. In conclusion, the result of analysis indicated the weak-tie relation can increase the extra innovative concept.

Keywords: Interactive Evolutionary Computing, Social Network Analysis, Chance Discovery, Grounded Theory, Data Mining.

1 Introduction and Literal Review

Simon proposed the process of decision making: intelligence, design, choice and review [7]. The decision maker has to search the environment to find out problems or decision chance as we know which scenario will unfold as intelligence. If we can identify these implications from all scenarios, we are confident that we are making much better robust plans. Thus, we want early warning signs, which tell us that these scenarios are beginning to unfold as intelligence.

1.1 How to Discover the Chance

The strong-tie network that the person of the strong relation formed is unable to bring the new information to us. It can improve this question in the weak-tie that Granovetter proposed [2]. However, the information of the weak-tie is unable to obtain by traditional data mining, must solve with chance discovery that Ohsawa proposed [4]. To avoid the risks of the decision making and misunderstanding, Ohsawa used Key-Graph and the double helical model to discover subtle but important "early warning signs" for making decisions [5]. It implied that human behavior was behind the data. The decision maker expressed the data structure to recognize the meaningful bridge based on his multi-social relationship. Therefore, we can classify the social relations

and can fit the data into multi-social networks before the data analysis, and then the decision maker can interact with the analyzed data to discover important chances.

1.2 How to Narrow Down the Innovative Data

In evolutionary computation (EC), the better chromosome has the best chance to propagate genes to the offspring; but in IEC [8], the interactive process not only supplies the stimulating information discovering what he wants, but also supplies the choosing and recombining power make a creative product. In addition, Hsu and Huang [3] said that appropriate search space can reduce fatigue problem. Wang, Sung and Hong [9] proposed On-Chance operator that can improve the quality of solution. In this paper, we decided follow the concept of value-focused thinking building our search space.

2 The Interactive Chance Discovery (ICD) Framework

Our framework is presented as Fig. 1 below:

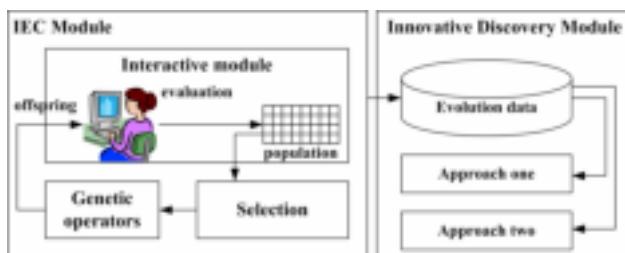


Fig. 1. The flow chart of research

2.1 Extracting Preferable Creative Ideas: Brainstorming and IEC

Step 1: Creative Ideas Generation

The brainstorming is a creative technique for generating the ideas to solve the problem within short time period [6]. In this study, we collected those ideas to design the attribute and attribute levels of cellular phone, and then encoded it.

Step 2: Narrow Down the Useful Data

It is important to discriminate between noisy and useful from evolution data. Therefore, we are choosing the highest one from individual evolution data.

Step 3: Data pre-processing

The score present user's preference. Thus, we weighted the evolution data according to the chromosome's score. For example, if a chromosome's gene and score is G-P-R-S and 3 respective. Then, the G-P-R-S will copy three times as "G-P-R-S G-P-R-S G-P-R-S" into new data set.

2.2 Innovative Discovery Module

We introduced two indexes to measure the high possibility innovative chance.

2.2.1 Approach One: To Discriminate Strength between Strong and Weak from Homogenous Characteristics

a. Centrality index:

There are three measures of centrality that are widely used in network analysis: degree centrality, betweenness, and closeness. Thus, if weak node (or term) v connected many strong-ties, then its degree and value would not small. The degree centrality which was defined as the number of links incident upon a node could be used to measure its degrees. The betweenness index was a centrality measure of a vertex within a graph; for example, for a graph $G:=(V, E)$ with n vertices, the betweenness $C_B(V)$ for v was:

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (3)$$

where σ_{st} was the number of the shortest geodesic paths from s to t , and $\sigma_{st}(v)$ was the number of the shortest geodesic paths from s to t that passed through a weak term v . The closeness index was useful to measure the mean geodesic distance between a weak term v and all other terms reachable from it:

$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v, t)} \quad (4)$$

where $d_G(v, t)$ was the distance between weak term v and term t . The weak node (or term) was like a bridge, and betweenness and closeness were used to measure its bridge's effect.

b. Cooperating index:

In this paper, we treated degree centrality, betweenness, and closeness as compound index named cooperating that used to measure the resources which node owns in the network. This index also used to measure the strength of helping grow each other. Therefore, the strong-tie set was removed according to the centrality index, and then cooperating index was used to discover an additional relationship between nodes.

2.2.2 Approach Two: To Find High Possibility Innovative Chance from Heterogeneous Characteristics

If a node (or term) v has high cooperating index value and connected to cluster that has high centrality index value, and then the node (or term) v could be innovative.

3 Experiment and Discussion

3.1 Experiment Design

We followed value-focused thinking to establish all possible solution space. Thus, we held brainstorming session to gather the customer values. There are 75 people were invited to participate in the brainstorming session. All participants were college and MBA students. The brainstorming session held during summer 2004. At brainstorming session, we asked participants to write down their possible thinking when purchasing cellular phone. Once the brainstorming session stopped, we can create an initial set of attribute levels according to the result of brainstorming session. The chromosome structure and the phenotype are shown in Fig. 2.



Fig. 2. The system interface of IEC

According to the concept of Takagi proposed [8], the score was related to the distance between the practical product and the preferable product of user's mind. Goldberg thinks that can produce innovation through the operation of EC [1]. Therefore, collected IEC evolution data could help us discover many useful innovations. The parameter of evolving process is set up as follows: population size is 6; selection method is wheel method and strategy is elitism; crossover rate is 0.8; mutation rate is 0.01.

3.2 Experimental Results

We developed an IEC-based social network system to analyze IEC evolution data. In the system, there are different symbol for represents different meanings. For example, the star represents high degree, high betweenness, and high closeness; the sun represents high degree, the bridge presents high betweenness, and so on. A series of try and error to determine the threshold settings of the index mentioned earlier. Then, we sifted the evolution data to find core set that would help us. Fig. 3 illustrates symbol status.

As the Fig. 4 indicates, the core set were “Audio Player”, “Network”, “Camera”, “Connectivity”, “Screen Colors”, and “Screen Size”. And the potential innovations are “Video Player”, “Games”, and “Document”.

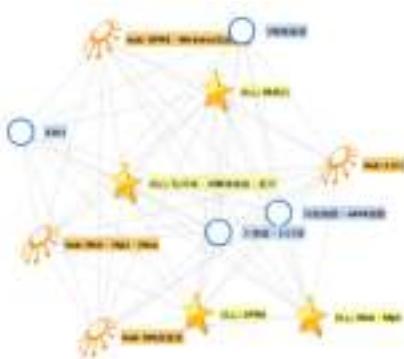


Fig. 3. An example of IEC-based social network system



Fig. 4. Approach 1: Sifted the core set

4 Conclusion

The contribution of this paper is to point out the importance of human's knowledge and social relationship between strong and weak. We proposed an ICD framework for innovation discovery. Experimental results and technology grow indicated the operator can help us to identify the possible scenario in future. The ICD framework is simply, however, and there are many operations, such as how to create customer values, how to integrate all of the values into the variable space, etc., that need further refinement.

Reference

1. Goldberg, D.E.: *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Addison-Wesley, Reading (2002)
2. Granovetter, M.: The Strength of Weak Ties: A Network Theory Revisited. *Sociological Theory* 1, 201–233 (1983)
3. Hsu, F.C., Huang, P.: Providing an appropriate search space to solve the fatigue problem in interactive evolutionary computation. *New Generation Computing* 23(2), 114–126 (2005)
4. Ohsawa, Y., Benson, N.E., Yachida, M.: KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor. In: Proc. Advanced Digital Library Conference (IEEE ADL 1998), pp. 12–18 (1998)
5. Ohsawa, Y.: Modeling the Process of Chance Discovery. In: Ohsawa, Y., McBurney (eds.) *Chance Discovery*, pp. 1–15. Springer, Heidelberg (2003)
6. Osborn, A.F.: *Applied imagination: Principles and procedures of creative problem solving*, 3rd edn. Charles Scribner's Sons, New York (1963)
7. Simon, H.A.: *The New Science of Management Decision*. Harper & Row, New York (1960)
8. Takagi, H.: Interactive evolutionary computation: fusion of the capacities of EC optimization and human evaluation. *Proc. IEEE* 89(9), 1275–1296 (2001)
9. Wang, L.H., Sung, M.Y., Hong, C.F.: Interactive evolutionary computation framework and the on-chance operator for product design. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2006*. LNCS, vol. 3907, pp. 565–574. Springer, Heidelberg (2006)

Evolving Approximate Image Filters

Simon Colton and Pedro Torres

Computational Creativity Group
Department of Computing, Imperial College London
{sgc,ptorres}@doc.ic.ac.uk
<http://www.doc.ic.ac.uk/ccg>

Abstract. Image filtering involves taking a digital image and producing a new image from it. In software packages such as Adobe’s Photoshop, image filters are used to produce artistic versions of original images. Such software usually includes hundreds of different image filtering algorithms, each with many fine-tunable parameters. While this freedom of exploration may be liberating to artists and designers, it can be daunting for less experienced users. Photoshop provides image filter browsing technology, but does not yet enable the construction of a filter which produces a reasonable approximation of a given filtered image from a given original image. We investigate here whether it is possible to automatically evolve an image filter to approximate a target filter, given only an original image and a filtered version of the original. We describe a tree based representation for filters, the fitness functions and search techniques we employed, and we present the results of experimentation with various search setups. We demonstrate the feasibility of evolving image filters and suggest new ways to improve the process.

1 Introduction

Image filtering is the process whereby a digital image is taken as input and a new image is produced, usually via a mathematical transformation of the bitmap information in the original image. Such image filtering finds application in numerous areas, including machine vision, medical imaging [1], graphic design and the visual arts. In the latter two cases, sophisticated software packages such as Adobe’s Photoshop are employed to produce artistic versions of images, usually via a chain of image filter applications. Such software often includes hundreds of different image filtering algorithms, with each having many parameters to explore the space of filters available. While this freedom of exploration may be liberating to artists and designers, the plethora of opportunities for less experienced users may be daunting. While Photoshop provides image filter browsing technology, it does not yet enable image filter search, i.e., retrieving a filter given an example of an image produced by it, nor can it construct a filter from scratch to approximate a target image filter. We investigate here whether it is possible to automatically evolve an image filter to approximate such a target filter, given only an original image and a filtered version of that original.



Fig. 1. Example image filter tree consisting of transforms in blue circles: A=Add Colour, C=Convolution, I=Inverse, M=Median and T=Threshold; and compositors in red squares: A=And, F=Fade, M=Min and O=Or. Image inputs are in green diamonds. An example original image and the filtered version are shown.

We look here at the general question of evolving a filter to approximate a given target filter. As described in section 2, we have developed a fairly expressive tree-based representation of image filters, and we have built a library of 1000 image filters in this representation. As described in section 3, filters represented in this way are amenable to crossover and mutation, and hence admit an evolutionary search. As described in section 4, we experiment with two ways in which to seed an initial population, namely by retrieving filters from the library, as per [9], and by random generation. We also experiment with four fitness functions, and discuss the visual differences between the types of filters that are evolved using them. Working with 75 image filters, 46 of which are from Photoshop, and 29 from our library, we show that, while it is difficult, automatically evolving suitable approximations to filters is feasible, but there is much room for improvement. We conclude by describing some of the planned improvements.

2 Representing Image Filters

We represent image filters as a tree of fundamental (unary) image **transforms** such as inverse, lookup, threshold, colour addition, median, etc., and (binary) image **compositors** such as add, and, divide, max, min, multiply, or, subtract, xor, etc. As an example, the *Median* transform takes two parameters, the first of which determines the way in which the median RGB values of each pixel is calculated, and the second of which determines the extent of the neighbourhood that is taken into account when the median is calculated. As an example compositor, the *And* compositor takes two images and calculates the *and* logical value of the pair of bitmap RGB values at each coordinate in the image. An example tree is provided in figure 1, where the overall filter uses seven transform steps and six compositor steps, and the original image is input to the tree seven times.

As described in subsection 3.2 below, image filters can be generated randomly. Each time an image filter generated in this random way produced an interesting visual effect on one or more original images, it was added to a library of filters, until we had built up 1000 such filters, categorised into 30 sets of filters, roughly



Fig. 2. Ten filters applied to the original image from figure 1

according to how the filtered images look, e.g., there are categories for filters which are blurred, grainy, monotone, etc. The time taken to apply a filter is roughly proportional to the size of the input image multiplied by the size of the tree. Over the entire library of filters, the average number of nodes in a tree is 13.62, and the average time¹ to apply a filter to an image of dimension 256 by 384 pixels (the size of images in the standard Corel library) is 410 milliseconds. To give an indication of the visual variety achievable by the filters in the library, in figure 2, we provide 10 filtered versions of the original image from figure 1.

3 Evolving Filters

We are addressing the following problem: given an image I and a target filtered version of I , which we denote T_I , construct a filter F such that the filtered image, F_I , gained from applying F to I approximates T_I , to a good standard, as evaluated by a given fitness function. In order to describe these techniques, we first discuss the fitness functions we used, followed by how we randomly generate, crossover and mutate filter trees.

3.1 Fitness Functions

As with most evolutionary search applications, correctly measuring the fitness of individuals is key to success. In our case, the fitness function will be used to order individual image filters in terms of how close their output is to the target filter's output when applied to a given original image. We envisage (at least) the following two reasons to evolve image filters. In the first scenario, a user has found an example of a filtered image and wants to construct a filter to approximate it as closely as possible. In the second scenario, a user has an original image and has found a filtered version of it that pleases them, but would like to see similar variations on the theme.

¹ On a Mac OS X machine running at 2.6Ghz.

To model the former case, we use the following fitness function:

$$1 - \left(\frac{\sum_{1 \leq x \leq w} \sum_{1 \leq y \leq h} (dist(T_I(x, y), F_I(x, y)))}{w * h} \right)$$

where w and h are the width and height of the images being produced respectively, and $dist(T_I(x, y), F_I(x, y))$ is the Euclidean distance in RGB-space between the pixel at point (x, y) in the target filtered image and the corresponding pixel in the image output by the filter being evaluated. Hence, if a filter achieves a score of 1, it perfectly reproduces the target filtered image at the bitmap level.

In the latter case, recreating exact pixel values is less important than gaining an overall visual style in the evolved filters. For this reason, we use two common features of an image, namely its colour distribution, and how grainy it is (i.e., the level of contrast in the image). We use the following weighted sum of these two features as a fitness function:

$$w_g * (1 - |g(T_I) - g(F_I)|) + w_c * (1 - diff(hist(T_I), hist(F_I)))$$

where $g(X)$ approximates the contrast² present in an image by measuring the average Euclidean distance in RGB-space of a pixel from its neighbours in image X and $diff(hist(T_I), hist(F_I))$ denotes the difference between the colour histograms of the target and evolved filtered images. This is measured by calculating the average of the difference over all the bins in a 4 by 4 by 4 colour histogram for both images. As we see in section 4, we experimented with three different pairs of weights $\langle w_g, w_c \rangle$ for the weighted sum.

3.2 Random Generation of Filter Trees

Filter trees can be randomly generated by starting with an input image node (a diamond-shaped node in figure 1), and iteratively choosing to replace an input image node with a transform node or a compositor node. If the node is replaced by a transform, then a single input image node is added to the tree above it, and similarly, if a node is replaced by a compositor, a pair of input nodes are added to the tree above it (as input to it). Each time a new node is generated, random parameters for that node are generated, with the parameters being within a suitable range of values. There are various ways in which we can control the random generation, including limiting the number of transforms/compositors/nodes and the longest branch in the tree, and controlling the likelihood of each particular transform/compositor being chosen as a node in the tree.

3.3 Crossover and Mutation

Representing image filters as trees enables both the crossing over of branches into offspring from each of two parents, and the mutation of trees, thus enabling

² We acknowledge that there are other methods for measuring the contrast in an image, in particular by taking the second order derivative of its colour histogram.

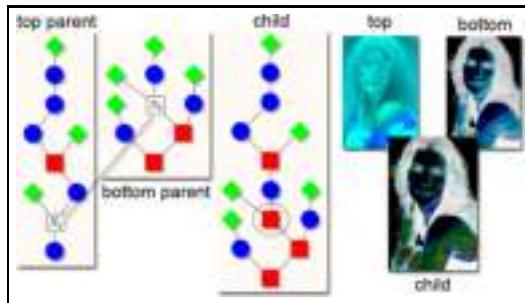


Fig. 3. An example of the crossover operation. We see that the child image inherits the colours from both parents and the texture from the top parent.

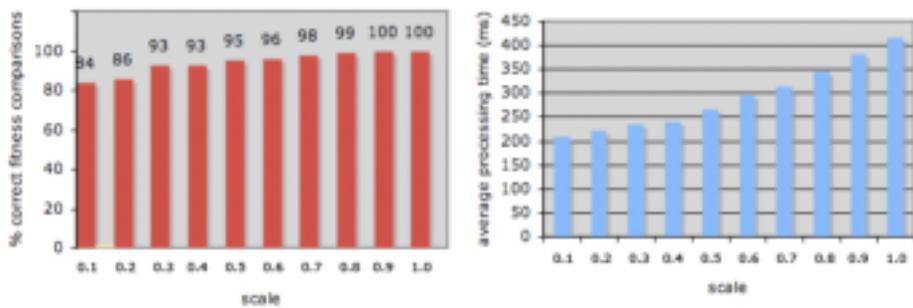


Fig. 4. First graph: percentage of correct fitness comparisons at various scales (from 250 tests). Second graph: average time taken to perform a single image filter and evaluation operation.

an evolutionary approach to filter construction. To perform crossover, we start with two parent trees (called the *top* and *bottom* parent), and we choose a pair of nodes: N_t on the top parent and N_b on the bottom parent. These are chosen randomly so that the size of the tree above and including N_t added to the size of the tree gained from removing the tree above and including N_b from the bottom parent is between a user-specified minimum and maximum. After 50 failed attempts to choose such a pair of nodes, the two trees are deemed incompatible, and a new couple is chosen. If they are compatible, however, the tree above and including N_t is substituted for the tree above and including N_b to produce the offspring. An example crossover operation is shown in figure 3.

We have implemented three techniques for mutating the offspring trees. Firstly, each node in a tree will be parameterised by a set of values, and the *low-mutate* mutation method looks at each parameter of each node and, with a probability of P_l , generates a new set of parameters for the node. Secondly, the *mid-mutate* mutation method looks at each node in the tree, and with a probability of P_m , chooses to replace that node by a suitable alternative (i.e., transforms are replaced by transforms and compositors are replaced by compositors). Finally, the *high-mutate* mutation method looks at each node in the tree

and with a probability of P_h replaces the tree above and including the node with a randomly generated sub-tree (as per the random generation method described above, and subject to the same restrictions as for entire filter trees).

4 Experiments and Results

Our ambition here is modest: we wish to show that evolutionary search for image filters converges on solutions which have some visual similarity with the target filter. We have only experimented with a very small subset of the parameters used in automating the evolution of image filters, and we discuss further experiments that we plan to undertake in section 6.

4.1 Reducing Image Size

As previously mentioned, the time taken to perform a filter is proportional to both the image size and the size of the filter. Hence, since in initial studies we saw no discernable difference in the quality of the filters produced using size-reduced images, we experimented with reducing the size of the original (unfiltered) image in order to increase the efficiency of the evolutionary process. Reducing the size of the image to be filtered is not detrimental to an evolutionary search if the ordering of individuals in terms of the fitness function does not change too much. To determine an appropriate scaling, we used the first fitness function described above, namely the average Euclidean distance of pixels in RGB-space. Scaling the original image at scales of $0.1, 0.2, \dots, 1.0$, in each case, we tested 250 sets of three filters f_1, f_2 and f_3 randomly chosen from our library of 1000 filters. For each triple, using f_1 as the target filter, we measured the fitness of f_2 in terms of its approximation of f_1 and the fitness of f_3 in the same terms. We repeated this for the scaled images, and recorded a hit if the same filter (from f_1 and f_2) was determined to be fittest in the scaled case as in the unscaled case. We also recorded the average time taken to perform a single filtering operation and fitness evaluation. The percentage hit rates and the average times are shown in figure 4. Note that these tests were performed with an original image of size 256 by 384 pixels on a Mac OS X 2.6Ghz. Looking for a balance between gain in efficiency and fidelity of fitness comparison, we chose to reduce our original image to 0.3 times its size in all subsequent experiments. This gives a 43.5% increase in efficiency for only a 7% loss in the fidelity of fitness comparisons.

4.2 Search Setups

We have barely scratched the surface of possibilities for the evolutionary search for image filters – indeed, our search setups are rather simplistic at the moment. Common to all the search setups are the following aspects: we used a population size of 100 and evolved for at most 100 generations. Moreover, the search was set to terminate if there was no improvement in the fittest individual or the average fitness over the population for five generations in a row. Note that in

all but a handful of the 375 sessions we ran, the search terminated before the full 100 generations were exhausted. We used a simple selection method: the top 25% of filters were put into an intermediate population, from which pairs were chosen randomly and a single offspring produced if possible. Compensating for discarding 75% of each population, to increase diversity, new filters were only allowed into the next population if the image they produced differed in at least 5% of its pixels with the images produced by all the filters which were already added. We found that in most cases, this measure was enough to avoid premature convergence of the search. Offspring filters were required to have between 3 and 20 nodes in their trees, and we imposed a mutation rate of 0.1 for each node in the offspring tree, using the mid-mutate technique only. We experimented with five search setups, which differed as follows:

- *Setup 1*: Random generation of the initial population, with a random tree size limit of 20 nodes. Fitness function: Euclidean RGB-distance.
- *Setup 2*: As setup 1, except the initial population is generated by searching through our library of 1000 filters using methods described in [9] to find the closest matching filters. Naturally, when the target filters were themselves from the library, we did not allow them to be retrieved into the initial population.
- *Setups 3, 4 and 5*: As setup 1, but with weighted histogram/contrast fitness described in §3.1 with weights $\langle 1, 0 \rangle$, $\langle 0.75, 0.25 \rangle$, $\langle 0.25, 0.75 \rangle$, respectively.

We experimented with 75 image filters, namely the 46 filters which come with the standard Photoshop distribution, using the default settings, and 29 from our library of 1000 filters, chosen to cover a wide range of visual styles. The fitness of the best individual seen in any population, averaged over all filters (both from our library and from Photoshop), for search setups 1, 2, 3, 4 and 5 was 0.92, 0.92, 0.94, 0.94 and 0.97, respectively. Hence, for all the search setups, it was possible to evolve filters with fitness scores greater than 0.9 on average, which is encouraging. However, this belies the fact that filters can be evolved to maximise the fitness functions without necessarily producing images which look like the target filter – see the next subsection for examples. We also note that choosing to generate filters randomly, or choosing them from our library produces very little difference in terms of the fitness of the best evolved filter. On inspection of the data, we found that – in general – the initial populations for setup 2 score worse than for setup 1, but the subsequent populations catch up fairly quickly. We also see that it is easier to evolve filters which maximise the histogram/contrast fitness function than the RGB-distance fitness function. The production of a new population takes around 73 seconds (on 2.6Ghz machine). Over the 375 sessions, the shortest/average/longest sessions produced 11/31/100 populations, taking 13 minutes, 37 minutes and approximately 2 hours respectively.

4.3 Illustrative Examples

In figure 5, we present 18 illustrative examples of the best (in terms of the appropriate fitness function) evolved filters, with 9 where our library provided



Fig. 5. Illustrative examples of the best filters evolved for each search setup. First column: target filters from our library. Second column: target filters from Photoshop.

the target filter and 9 where it was provided by Photoshop. We have tended to show the better results, but we include the approximations of the *embossed7* filter as an example where none of the search setups produced a visually similar or appealing filter. This example indicates a phenomena we saw fairly often, namely filters evolving to produce single colour filters (which minimises the RGB-distance on average when the target filter produces images dominated by one colour). Another phenomena we saw fairly often was the evolving of filters which produce rather greyscale images when the target filter actually produces quite colourful images. Again, this can be explained, as greyscale images have pixels which are not at extremes in the RGB-space, hence they tend to maximise fitness. Such a greyscale image was produced with search setup 2 for the *modern12* filter and to a lesser extend with setup 3 for *colourful14*. We describe some ways in which to combat this phenomena in section 6. Note, however, that when using

the RGB-distance fitness function, colourful filters can be evolved for colourful target filters, as evidenced in searches 1 and 2 for the *psychedelic13* filter.

Some highlights (judged entirely subjectively by ourselves) include: (a) the near perfect re-creation of the *bitone3*, *bright2*, *colourful14* and *psychedelic13* filters (and also the *grey1* and *vivid1* target filters, not shown in figure 5) (b) the fairly good approximation of some Photoshop filters, including the *charcoal*, *dark strokes*, *diffuse glow*, *film grain*, *glowing edges*, *neon glow*, *patchwork* and *stamp* target filters. Also, many filters found using the histogram/contrast fitness function were not exact matches, but had a similar visual style to their targets, e.g., the filter developed using setup 5 for *fade4* captures its style well; the filter generated by setup 2 to approximate *painterly28* has a particularly painterly style, and while the approximation to the Photoshop *chrome* filter produced by setup 5 is a bad approximation, it does capture some of its essence. On the whole, the evolution performs better for our library filters than for the Photoshop filters. This is largely due to the Photoshop filters being rather subtle, i.e., they alter the image very little. In these cases, an identity filter scores highly for all the fitness functions, and so the search tends to discover ways in which to construct trees which do nothing. However, there are some exceptions: e.g., the *accented edges* filter is very subtle, but setups 1 and 4 found good approximations.

5 Related Work

The NEvAr system [3] is an evolutionary art tool which uses a similar tree-based representation for image filters but is different in that the system described here (1) explicitly uses an initial existing image as a seed for evolution, (2) uses tree nodes which correspond to high-level transformations of the input images rather than compilable code, (3) uses a target image to derive a fitness function. A general approach for evolving image filters tree was introduced by Sims [7] and although complex image transformation were allowed at node level, it still differs from the work presented here on aspects (1) and (3) above. In [4], which is conceptually close to the approach described here, the authors use Genetic Programming to evolve programs that give colour to greyscale images based on existing coloured training images. Particular applications of image filter evolution include evolving noise removal filters for graphics processor units using Cartesian genetic programming [2] and an implicit context representation [8], and evolving simple filters such as the salt and pepper filter for Field Programmable Gate Arrays [6].

6 Conclusions and Future Work

Using a tree-based representation of image filters, we have investigated how to evolve approximations of a target filter, by referring to an image produced by the target filter when calculating the fitness of individuals in a population. We have experimented with five search setups which use four fitness functions, and we have shown that in some cases we can evolve filters which very closely approximate the target, and in other cases, we can evolve filters which have a similar

visual style to the target. On inspection, 17 of the 46 Photoshop filters (37%) were approximated in an appropriate way by at least one of the search setups, and 21 of the 29 filters from our library (72%) were adequately approximated. While this is encouraging, there is certainly room for improvement.

In future work, we hope to find more effective search setups, by experimenting with more sophisticated selection techniques, as well as parameters such as the population size, mutation rate, mutation style, and different methods for tree-based crossover [5]. We also plan to experiment with different fitness functions, to address some of the limitations identified here. In particular, searches can converge on identity filters, filters which produce single-colour images and greyscale images (for colourful target filters). To address this, we plan to experiment with fitness functions which look at the mapping from the original to the filtered image that the filter produces. In particular, we will use a sliding scale of importance for pixels: those which are not altered at all by the target filter will be ignored by the fitness function, while those which are altered a great deal will be used prominently. We hope that such a fitness function will also enable the evolution of filters which approximate the more subtle Photoshop filters, where most of the pixels remain the same. We will also consider different numbers of generations of consecutive identical fitness as a terminating criterion for evolution. Finally, given the lack of perceptual uniformity in RGB space, we plan to carry out the same set of experiments for different colour spaces, e.g. HSV.

By substituting the transforms and compositors in our filter trees by more sophisticated operations, we believe that evolving filters could be a very useful tool in an environment such as Adobe Photoshop: inexperienced users could ask the software to produce a filter using only an example image, and experienced users could ask the software for variations on a theme. We believe that future graphic design software should be able to act as such a creative collaborator, and we hope to see AI methods being used increasingly in this field.

Acknowledgements

We would like to thank Stefan Rüger and João Magalhães for their valuable suggestions regarding image analysis and the anonymous referees for their useful comments. This work is supported by EPSRC grant EP/F067127.

References

1. Behrenbruch, C., Petroudi, S., Bond, S., Declerck, J., Leong, F., Brady, J.: Image filtering techniques for medical image post-processing: an overview. *British Journal of Radiology* 77(2)
2. Harding, S.: Evolution of image filters on graphics processor units using Cartesian genetic programming. In: IEEE Congress on Evolutionary Computation (2008)
3. Machado, P., Cardoso, A.: All the truth about NEvAr. *App. Int.* 16, 101–118 (2002)
4. Machado, P., Dias, A., Duarte, N., Cardoso, A.: Giving Colour to Images. In: Proceedings of the AISB 2002, Symposium on Artificial Intelligence and Creativity in Arts and Science (2002)

5. Poli, R., Langdon, W.: Schema Theory for Genetic Programming with One-Point Crossover and Point Mutation. *Evolutionary Computation* 6(3), 231–252 (1998)
6. Sekanina, L., Martínek, T.: Evolving image operators directly in hardware. In: Proc. of Genetic and Evol. Computation for Image Processing and Analysis (2007)
7. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of SIGGRAPH 1991 (1991)
8. Smith, S., Leggett, S., Tyrrell, A.: An implicit context representation for evolving image processing filters. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2005*. LNCS, vol. 3449, pp. 407–416. Springer, Heidelberg (2005)
9. Torres, P., Colton, S., Rüger, S.: Experiments in example-based image filter retrieval. In: Proceedings of the Cross-Media Workshop (2008)

On the Role of Temporary Storage in Interactive Evolution*

Palle Dahlstedt

Dept. of applied information technology, Chalmers University of Technology/University of Gothenburg, 41296 Göteborg, Sweden
and the Academy of Music and Drama, University of Gothenburg,
Box 210, 40530 Göteborg, Sweden
palle@ituniv.se

Abstract. In typical implementations of interactive evolution of aesthetic material, population size and generation count are limited due to the time-consuming manual evaluation process. We show how a simple device can help to compensate for this, and help to enhance the functionality of interactive evolution. A temporary storage, defined as a number of easily accessed memory locations for evolved objects, adjacent to the evolving population, can be regarded as a non-evolving extension of the population. If sufficiently integrated into the workflow, it provides compensation for limited genetic diversity, an analogy to elitism selection, and means to escape from stagnation of progress through backtracking and reintroduction of previous genomes. If used in a structured way, it can also help the user form a cognitive map of the search space, and use this map to perform a structured, hierarchical exploration. The discussion is based on experiences from a series of implementations of interactive evolution of music and sound, but should be relevant also for other forms of artistic material.

Keywords: Interactive evolution, aesthetic selection, temporary storage, workflow, cognitive maps.

1 Introduction

Interactive evolution is the predominant technique for evolving aesthetic objects, introduced by Dawkins [1] and first applied to aesthetic material by Sims [2]. It has since been successfully applied to the evolution of all kinds of artistic material such as images, designs, buildings, synthesized sounds and musical structures. The advantage of this method is that the fitness function is replaced by human aesthetic evaluation, since it is obviously very difficult to formalize evaluation criteria for aesthetic objects. There are several reasons. Often, one does not know what one is looking for, hence, the fitness criterion is unknown. Or, one may change one's preferences during the process, when something unexpected but very interesting is found. Sometimes, the preferences are implicit and vague,

* This work was partially funded by a grant from The Swedish Research Council.

and can only be formulated when a good solution is found — you do not know it's right until you see it. In all these cases, interactive evolution is a powerful technique for exploring an unknown space of possible solutions, in search for suitable artistic material.

However, interactive evolution has its disadvantages. Human evaluation takes time, which seriously limits the possible number of generations and the population size. This is what Biles called the fitness bottleneck [3]. This is especially difficult when working with time-based material, e.g., music and sound. Musical objects cannot be compared side by side, as images, but have to be auditioned in sequence. Limitations in our aural memory also limits the population size; we cannot keep a very large number of sounds in our head.

Since we are not talking about an optimization process, but an exploration, the limitations on generations and population may not be an obvious drawback. But the fact is that the search space is still vast, and we want to be able to search it as thoroughly and efficiently as possible.

The limitations on population size has other consequences, too. A small population is much more prone to genetic stagnation, since it cannot maintain any significant diversity in itself, and the small number of candidate solutions may not be sufficient to find a plausible next parent.

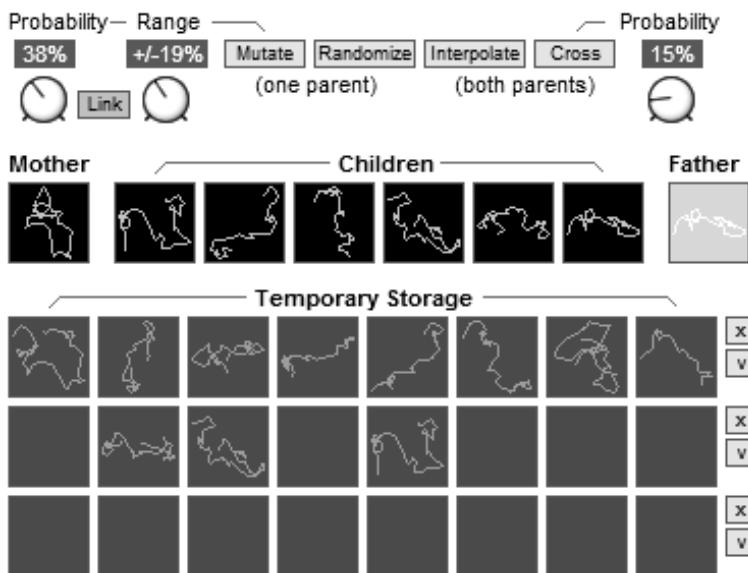


Fig. 1. Temporary storage as implemented in the user interface of the Patch Mutator system (part of the dsp development environment Nord Modular G2 [4]), used to evolve sound synthesis parameter sets. Placed directly beneath the active population, the storage can be subject to all the same operations as the active population in the upper row (parents and children): auditioning, move or copy, one-parent reproduction, two-parent reproduction. The two buttons to the right are used for clearing or saving a row of objects.

In this paper, I will discuss how a simple device can compensate for some of these disadvantages: a temporary storage, a scratch pad with easy access for temporary results found during the evolutionary exploration. It seems almost trivial, but if sufficiently integrated in the workflow, it brings a whole series of advantages to the interactive evolutionary process, and in spite of this, it has to my knowledge not been discussed much in the literature, and is not often implemented. Defined as a number of memory locations positioned side by side with the current population, with equally fast access for auditioning, saving and restoring, it can be considered as an extended, non-evolving part of the population. See Fig. 1 for an example. Such a temporary storage compensates for a small population, provides something equivalent to elitism selection, and enables backtracking in the search process. We will also see how it makes possible more complex hierarchical search methodologies, and helps the user to form and gradually refine a cognitive map of the search space.

The discussion is based on experience from design, development and regular professional use of a number of systems for interactive evolution of sound, music and scores, over a period of nine years — with and without a temporary storage (see, e.g., [5,6,4,7]). The different uses of the temporary storage and its advantages result from careful analysis of how the author used it intuitively over long periods of time, without being conscious of the actual underlying mechanisms. At one time, when developing a new implementation and omitting a temporary storage, its advantages became very obvious (by their absence). A more careful study was started, leading to this paper.

While musical applications form the empirical basis for this paper, the conclusions should be equally valid for implementations of interactive evolution of objects from other art forms, such as graphics, design, choreography or architecture.

A number of systems for interactive evolution of art have included databases of previously evolved objects, e.g., the works by Sims [2], Rooke [8]. Machado and Cardoso [9] also briefly discuss the significance of a database, and how reinsertion of objects from the database can help escaping from local minima. However, it is unclear to me if the databases in these implementations are integrated in the user interface with the same ease of access as the current population (e.g., a single click), which I believe is crucial to integrate it seamlessly into the workflow, and to achieve the advantages discussed in this paper. Despite its simplicity, temporary storage in such a directly accessible form is still lacking in many implementations of interactive evolution of artistic material. As an example, one recently released software synthesizer¹, based on the technique of aesthetic selection, only allows the user to save plausible sounds as named files, severely limiting the available search methodologies and slowing down the workflow.

Other solutions have been suggested to compensate for the inherent limitations of interactive evolution. Automated pre-selection of suitable objects or a combination of manual and automated fitness scoring [10] can allow larger populations, and small batches of generations can be processed between each manual evaluation pass, allowing a larger number of generations. Fitness for objects not

¹ SynPlant by SonicCharge, see www.soniccharge.com.

evaluated by humans can be estimated based on previous evaluations (see, i.e., [11], where a few different algorithms are compared). Another approach is to let the algorithm keep track of the difference between individuals, to avoid stagnation [12]. In [13] various approaches to automatic regulation of mutation rates are investigated. But all these techniques require at least some kind of formalized fitness evaluation or distance metric, which is not always possible to define. In this paper, the problem is instead addressed by giving the user a larger repertoire of operations, allowing more sophisticated search strategies.

2 Workflow

The focus of the author's research in this area has been on tools that not only work as experiments, but real-life tools for sound design and composition. In those situations, a stream-lined workflow can further minimize the time-consuming manual selection process, and make the difference between a cumbersome experimental tool and a mind-opening design environment. Obviously, auditioning and selection must be fast. Often, manual scoring of each object takes too much time, and hard choice selection is preferable, where objects to be used as parents are explicitly hand-picked in each generation by the user, and subjected to one- or two-parent reproduction according to the choice of the user. That is, the user decides explicitly to create a new generation from one selected individual (by mutations) or from two (by crossover). In the following discussion, this kind of hard choice selection is assumed.

Stream-lined workflow is also crucial for a temporary storage to be completely integrated in the selection process. A normal file save functionality is good for archiving of final results, but is too slow to be integrated in the selection process, or to be considered an extended part of the population.

The most obvious use of a temporary storage is to save promising temporary results at various stages in the exploratory process. These objects can be reintroduced into the breeding at any point in time, and can be subject to final selection and saved, when finished. A typical workflow is shown in Fig. 2.

During selection, the individuals of the current population are not only compared to each other or the current parent or parents during selection, but also (possibly unconsciously) to the sounds in the temporary storage, which can at any point be reintroduced into the breeding. In this way, the temporary storage really forms a part of the population, albeit non-evolving, and to a certain degree compensates for the small population size. Since the members of the temporary storage do not change with every generation (unless something is stored or deleted), they do not have to be auditioned very often to be remembered by the user, thus they provide a compensation for the small population size without adding much evaluation time.

3 Avoiding Stagnation

With a small population, the evolutionary process can get stuck at a local maximum. There is not room for much heterogeneity within the population, and it

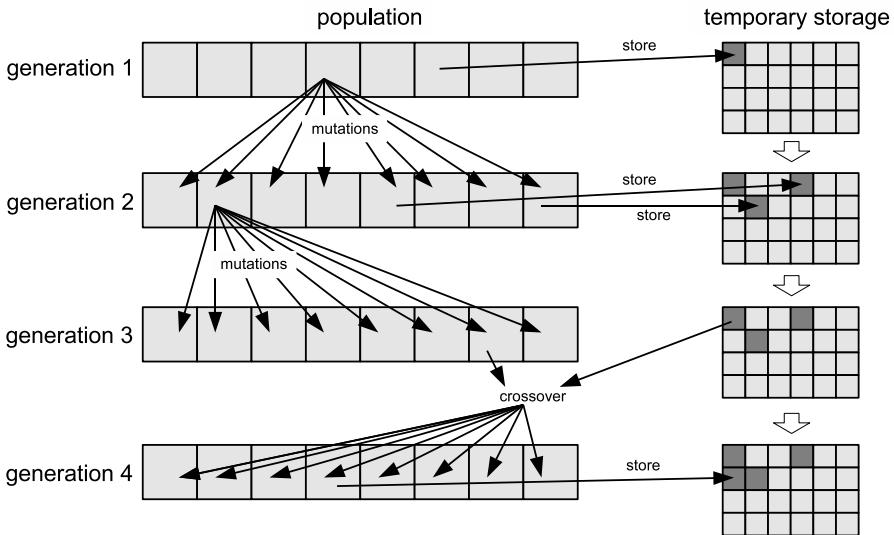


Fig. 2. A typical, but simplified, example of the workflow when using interactive evolution with hard choices and temporary storage. In the first generation, one interesting object is stored, and another one is used for reproduction by mutations, to form generation 2. Two from the offspring are stored, and a third one is used to produce the generation 3. Then, the next generation is created by crossover from one object from the active population and one from the temporary storage. From generation 4, one object is stored. Normally, the process goes on over many more generations, but the example shows the typical operations involved.

is up to the user to detect when the process is stuck. Here, it is worth pondering if an exploratory process can get stuck in the same meaning as an optimization process — with an undefined fitness function, there may be no maximum or minimum. But Livingstone did not explore the jungle to find better places, but to find *new* places. If he got stuck in a dead end, he stepped back and found another path. The stagnation is not in the lack of quality of the current objects, but in the halted traversal, in the lack of movement.

The user can reintroduce some diversity by temporarily increasing the mutation rate, or by reintroducing fresh genes from the temporary storage. In this context, it is worth mentioning that crossover within a very small population of one or two-parent offspring does not make much sense, since the objects will be very similar, and the population will gradually converge. However, crossover between an individual from the population and one from the temporary storage, or between two from the storage, can be a fruitful way of reintroducing new genes into the population.

Another way to handle genetic stagnation, is to use the temporary storage for backtracking. By retreating to a recently stored object, evolution can be continued afresh from there. Backtracking can also be achieved through regular multi-step undo, which is available in some implementations. But undo is a

stack-based process, and history cannot be traversed in arbitrary order, limiting its use as a navigational tool.

In practice, the temporary storage also acts in a way analogous to the elitism technique often used in genetic algorithms, i.e., to retain some of the best individuals from each generation unaltered into the next generation to avoid losing attained fitness. When the user selects a specific object for the temporary storage, it is promoted to the special status of not being affected by genetic operators (e.g., mutations), but still being part of the (extended) population, since the stored objects are implicitly taken into account during the aesthetic evaluation. Thus, the temporary storage provide a mechanism similar to elitism.

4 Search Methodologies

With access to a temporary storage, the search space can be navigated in very refined ways.

One approach is to first get an overview of the space, or at least a good sample of its potential, by doing a random search for a set of suitable seeds, which are put in the temporary storage. Then, the vicinity of each seed can be explored, first in large steps, then gradually smaller, zooming in (by decreased mutation rate).

The subspaces between the seeds can be examined in much the same way, through initial crossover and gradually smaller mutations, with found gems saved to the temporary storage, to be used in next iteration of what could be regarded as a divide and conquer approach. This alternation between wild exploration and careful refinement can be compared to the 'muse' and 'editor' in Nachmanovitch's model of creativity [14], exploiting the uncontrolled randomness and let its result be subject to refined control.

The temporary storage can facilitate selection when the user is unsure about specific objects. If the judgment is undecided, the object can be stored temporarily, to postpone the decision until further alternatives have been auditioned. Similarly, interesting objects that cannot currently be used as parents can be saved for later exploration; not all branches can be traversed at the same time. In this way, the temporary storage compensates for the rigidity of hard choice selection, allowing softer choices when needed.

5 Charting the Space

The temporary storage may also act as a memory, a trace of previous exploration. Livingstone did not traverse an unknown continent without charting his path, thus unable to retrace found treasures. In the process of breeding, stored objects are like flagpoles, marking found gems for future refined exploration. When examined further, this storage of trajectory waypoints can help the user to form a cognitive map of the vast search space (see Fig. 3). These known points are placed in the spatial grid of the temporary storage (or whatever layout is used), in a pattern unlikely to correspond to the actual topology of the space. But this does not matter, because it is a map of the part of the space we *know*,

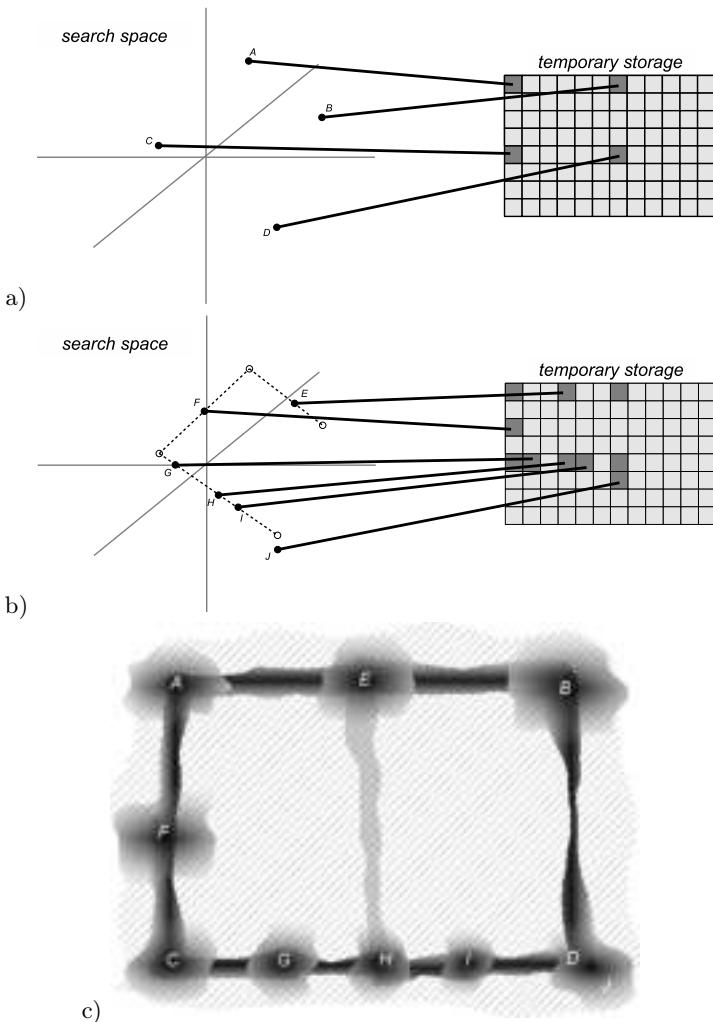


Fig. 3. The temporary storage helps the user create a cognitive map of the search space. *a)* Four points (*A*, *B*, *C* and *D*) are found and stored in the temporary storage. Their spatial distribution in the storage does not have to correspond to their actual relationships in the search space. The actual topology of the space is supposed to be too complex to grasp. *b)* Further points in between (*E*, *F*, *G*, *H* and *I*, found by crossover of *AB*, *AC* and *CD*) and around (*J*, found by mutating *D*) the known points are found and stored in the temporary storage, giving the user more detailed knowledge about the space, and the subspaces between the previously stored points. In reality, results of crossover are not located on straight lines between the parent points, but rather corners of a multi-dimensional rectangle defined by the parents, but this is a greatly simplified image of a high-dimensional space. *c)* A sketch of a cognitive map, showing the users knowledge (so far) of the search space. Darker areas are better known. The immediate surroundings of a known point are quite well known, since they are supposed to be similar to the stored point. Something is also known about the region between two points, but much less. The shaded area surrounded by the known points is vaguely known, since something can be guessed about crossovers of arbitrary pairs of known points.

and all maps we could possibly form in our minds have to be simplifications. The actual spatial relationships between stored points are likely to be too complex to visualize (we are talking about very high-dimensional spaces, or even infinite-dimensional spaces).

The map is gradually refined as the search progresses. We can envision the immediate surroundings of each stored point, if the space is sufficiently smooth. We also know something about the subregion between two points, which is possible to investigate by crossover and refined search, putting new flagpoles on the map in a hierarchical, gradual search process, recursively dividing the space in smaller regions. More effort is naturally spent exploring areas containing interesting results. Step by step, we form a cognitive map of parts of the space, with higher resolution in interesting areas, and this mental map corresponds directly to the stored points in the temporary storage, allowing efficient use of it to guide further exploration. The actual topology of the space is supposed to be too complex to grasp. We form the map of and around the points we know, and any new points outside this known region are “discovered”, further enhancing the creative nature of exploring an unknown space.

6 Discussion

Early implementations by the author arranged the temporary storage as a scrollable sequential list, while current implementations present the temporary storage place-holders in a rectangular grid of typically 3x8. This spatial arrangement seems to facilitate the use of the storage as a cognitive mapping device, but the size is still somewhat too limited for extensive exploratory sessions. The process has to be divided into several rounds, and the contents of the temporary storage have to be filtered, saved and cleared quite often to make room for new material. It would be interesting to experiment with a larger temporary storage, possibly of a more complex internal structure. One could imagine a grouped storage divided into tabs, with one active tab at a time, or a hierarchical structure, to better correspond to the structure of the actual search process. However, if the storage becomes too large or too complex, access to individual objects for audition and breeding will be slower, and the storage will lose its role as an natural extension of the population.

An interesting issue is how the objects could be visually represented in the storage. Images can always be represented by thumbnails, but for sounds and musical structures, the question is not trivial, and there are many possible graphical representations, depending on the kind of music. In some of our implementations, a turtle-graphics mapping of the synthesis parameters is used, forming a chromosome-like wiggly line, giving the brain a visual figure to associate to the aural memory. Piano-roll notation and spectrograms are also possible to use.

In contrast to the more common techniques of increasing the efficiency of interactive evolution (such as pre-selection), a temporary storage does not require any automated fitness function, and instead increases the influence and control of the user. Increase in efficiency is more in the sophistication of the available

exploratory search strategies than in the sense of saving time by doing hidden computations. Certainly, there is nothing that prevents a temporary storage to be combined with other efficiency-increasing techniques, when technically possible, to further improve search capabilities. This could be an interesting path for future investigations.

7 Conclusions

We have shown how a simple temporary storage can greatly enhance the usability and functionality of implementations of interactive evolution. If used actively, it can compensate for a small population size and the lack of diversity within the population. If used in certain ways, it can also provide the functionality of elitism selection, and be used as a basis for a cognitive map of parts of the search space, based on hierarchically structured search processes. In reality, all these techniques are used in conjunction, forming one advanced navigational, exploratory tool. The user does not have to think of it as specifically emulating elitism, avoiding stagnation or as a compensation for a lack of diversity. As mentioned, these ways of working were first developed spontaneously and intuitively by the author, and only later analyzed. Hence, no special knowledge is necessary to take advantage of them. But it is important to look at these techniques in detail, and to show that these well-known mechanisms have analogies in this seemingly simple system, which should make us feel more confident using it, and possibly lead to more implementations. With this knowledge, we can safely explore the unknown jungle for hidden treasures, the temporary storage being our combined logbook and gradually expanding map.

References

1. Dawkins, R.: *The Blind Watchmaker*. Longman Scientific and Technical, Harlow (1986)
2. Sims, K.: Artificial evolution for computer graphics. In: ACM SIGGRAPH 1991 Conference Proceedings, Las Vegas, Nevada, pp. 319–328 (July 1991)
3. Biles, J.A.: GenJam: a genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference, Aarhus, Denmark, pp. 131–137 (1994)
4. Dahlstedt, P.: Evolution in creative sound design. In: Miranda, E.R., Biles, J.A. (eds.) *Evolutionary Computer Music*, pp. 79–99. Springer, London (2007)
5. Dahlstedt, P.: Creating and exploring huge parameter spaces: Interactive evolution as a tool for sound generation. In: Proceedings of the 2001 International Computer Music Conference, Habana, Cuba, pp. 235–242 (2001)
6. Dahlstedt, P.: A MutaSynth in parameter space: Interactive composition through evolution. *Organised Sound* 6(2), 121–124 (2001)
7. Dahlstedt, P.: Sounds Unheard of: Evolutionary algorithms as creative tools for the contemporary composer. PhD thesis, Chalmers University of Technology (2004)
8. Cooke, S.: *Eons of genetically evolved algorithmic images*. In: Bentley, P.J., Corne, D.W. (eds.) *Creative Evolutionary Systems*. Morgan Kaufmann, San Francisco (2001)

9. Machado, P., Cardoso, A.: All the truth about NEvAr. *Applied Intelligence* 16, 101–118 (2002)
10. Sáez, Y., Sanjuán, O., Segovia, J., Isasi, P.: Genetic algorithms for the generation of models with micropopulations. In: Cagnoni, S., et al. (eds.) *EvoWorkshops 2003*. LNCS, vol. 2611, pp. 570–580. Springer, Heidelberg (2003)
11. Sáez, Y., Isasi, P., Segovia, J., Mochón, A.: An experimental comparative study for interactive evolutionary computation problems. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2006*. LNCS, vol. 3907, pp. 542–553. Springer, Heidelberg (2006)
12. Graf, J., Banzhaf, W.: An expansion operator for interactive evolution. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 798–802 (1995)
13. Breukelaar, R., Emmerich, M., Bäck, T.: On interactive evolution strategies. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2006*. LNCS, vol. 3907, pp. 530–541. Springer, Heidelberg (2006)
14. Nachmanovitch, S.: *Free Play: Improvisation in Life and Art*. Jeremy P. Tarcher / Penguin-Putnam Publishing, New York (1990)

Habitat: Engineering in a Simulated Audible Ecosystem

Alan Dorin

Centre for Electronic Media Art, Faculty of Information Technology,
Monash University, Clayton, Australia 3800
alan.dorin@infotech.monash.edu.au

Abstract. This paper introduces a novel approach to generating audio or visual heterogeneity by simulating multi-level habitat formation by ecosystem-engineer organisms. Ecosystem engineers generate habitat by modulation of environmental factors, such as erosion or radiation exposure, and provision of substrate. We describe *Habitat*, a simulation that runs on a two-dimensional grid occupied by an evolving population of stationary agents. The bodies of these agents provide local, differentiated habitat for new agents. Agents evolve using a conventional evolutionary algorithm that acts on their habitat preferences, habitat provision and lifespan, to populate the space and one another. This generates heterogeneous, dynamic structures that have been used in a prototype sonic artwork and simple visualisation.

Keywords: Ecosystem engineer, habitat, virtual ecosystem, artificial life, generative art.

1 Introduction: Fleas and Smaller Fleas

A short stroll through the Australian bush, even the local urban wetlands, reveals a coherent, spatialised soundscape generated by organisms and the interactions of biota and abiotia such as wind and water. Artificial life simulation can lend itself to electronic art that replicates the experience of such a stroll, by automatically generating complex, dynamic, heterogeneous patterns. In particular, virtual ecosystems appear perfect for this purpose and have been employed widely [1-4]. However, ecosystem simulation in artificial life has frequently leap-frogged aspects of ecology that are known to be responsible for generating much of the richness typical of real evolutionary systems [5]. This paper focuses on the idea of multi-layered habitat formation — habitat that appears directly on the bodies of organisms.

*So nat'ralists observe, a flea
Hath smaller fleas that on him prey,
And these have smaller fleas that bite 'em,
And so proceed ad infinitum — Swift, 1733*

The modern ecological terminology that encompasses this phenomenon is *ecosystem engineering* [6]. All organisms are physical ecosystem engineers to some extent. Physical ecosystem engineers alter the biotic or abiotic environment and thereby

control or modulate the availability of resources to (or forces acting on) other organisms. These physical changes destroy, maintain or create habitat for other organisms.

A tree is an example of a significant physical ecosystem engineer: it provides habitat for mosses on its limbs, insects under its bark and possums or birds in its hollows and branches; its roots trap soil and leaf matter, altering the impact of wind and water erosion; its branches harbour larvae or tadpoles within pools of rainwater and they provide shade and detritus for fungi. Coral produces reefs, wombats dig holes and worms break down vast quantities of leaf litter and other materials. These species have a massive impact on organisms around them, including of course themselves, their own species, and in particular, their offspring. Of course humans are extremely significant ecosystem engineers too. Ecosystem engineers are often *niche constructors*. That is, their impact on the environment alters the selection pressures acting on themselves and their offspring [7].

This paper does not investigate all aspects of ecosystem engineering. It focuses specifically on the production of habitat that is spatially coincident with the engineer organism. For instance, the tree just described is such a habitat provider. The dynamics of simulated habitat production are used to generate patterns for a sonic artwork and visualization.

1.1 Heterogeneity in Sonic and Visual Generative Art

The composition of pattern in sonic or visual phenomena is the very basis of generative art. Algorithms may easily generate boring repetition, or with some effort, near disorder. But when they create complex structure that is emergent from simple initial conditions and dynamical processes, generative artists are liable to sit up and take notice [8]. For this reason, cellular automata [9], L-Systems [10], ant-trail formation [11], and evolutionary algorithms [12] have all been employed. One of the goals of this paper is to generate coherent, dynamic, spatial complexity that may be experienced sonically and interpreted graphically. With any luck, the generated patterns will one day exhibit a richness associated with natural evolution and real biological habitat.

Existing works that have employed virtual ecosystems to generate soundscapes include: *Living Melodies* [2], which evolves singing organisms in a grid world; *Listening Sky* [1], a globe populated by musical organisms that is experienced via an eaves-dropper under user control; and *Eden* [4], a user-aware graphical and sonic installation of agents that forage and communicate their findings with one another audibly. The work *Plague* also has a sonic component generated from the interactions of evolving disease-ridden agents [13]. This paper employs a new technique, based on the simulation of habitat formation in an ecosystem, to achieve the goal of sonic spatiotemporal complexity and coherence.

2 Habitat, a Virtual Ecosystem Simulation

Ecosystem engineers must provide habitat that is sufficiently persistent if it is to come to support the evolution and continuation of species that depend on it. This simulation is designed to generate habitats provided by one organism for occupation by another. As in real ecosystems, the aim is to allow the emergence of habitat chains that piggy-back one another, supporting a range of niches.

In any real ecosystem, engineers may provide habitat for multiple dependent species (as does the tree described earlier) and the interactions between an engineer and its beneficiaries may be reciprocal. Although it is possible to extend the current simulation to permit this, our investigation leaves multiple habitat provision and reciprocal interactions as future work. An overview of the simulation appears below. Detailed parameter settings are discussed in section 3.

2.1 The Grid and Habitat Specification

The *Habitat* simulation runs on a toroidal grid of square cells. Each cell has a three-bit pattern designating the base habitat at its location. A different number of bits can be employed, however visualising the three bits as the colour components red, green and blue is effective and convenient. Any initial coloured pattern of bits across the grid cells can be established. In order to explore purely emergent structure in the absence of pre-specified heterogeneity, we generate our results from a grid with uniform white (1,1,1) base habitat.

2.2 Multi-layered Habitat Specification

Multiple agents may simultaneously occupy a *Habitat* cell by piggybacking on one another. Each agent has a genotype specifying the habitat bit pattern it *requires* and the habitat that it *provides*. The habitat provided always has fewer bits set than the habitat that an agent requires. I.e. to occupy a cell, an agent must use one or more habitat bits that represent “slots” in the environment. The habitat that an agent provides consists of the bits that remain after its required bits have been occupied (see Figure 1). For instance, for an agent to inhabit a white grid cell, its habitat preference bits must all be set: (1,1,1). Its habitat provision bits must then clear one or more of these bits, ensuring it provides habitat that is yellow (1,1,0), magenta (1,0,1), cyan (0,1,1), blue (0,0,1), green (0,1,0), red (1,0,0) or black (0,0,0). Agents are rendered as rectangles coloured according to the habitat they provide.

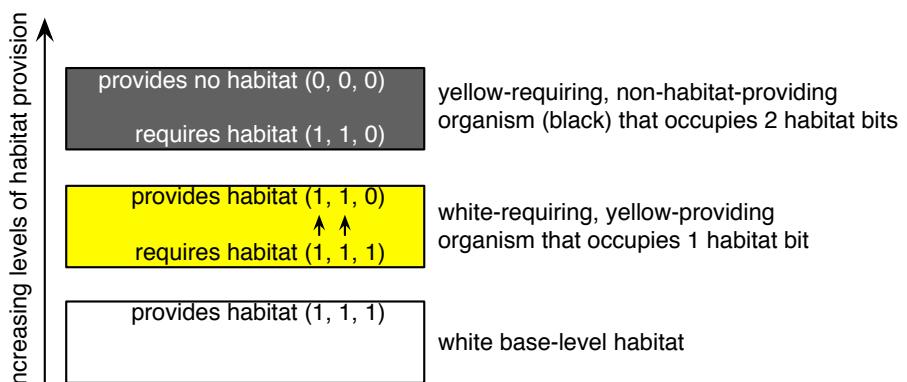


Fig. 1. A yellow agent (i.e. one that provides yellow habitat) occupies the white base and provides habitat for a black agent (i.e. one that provides no habitat itself)

An empty white cell first occupied by an agent that provides cyan habitat (0,1,1) may be occupied next by any agent with cyan preference bits. This second agent must then provide blue (0,0,1), green (0,1,0) or black (0,0,0) habitat, since it too must use one or more of the available habitat bits on the back of the cyan agent. Only a single agent type, black, can exist on top of a primary colour since there is only one bit of habitat remaining to be occupied. Hence, a complete habitat chain is always topped off with unusable black (Figure 2).

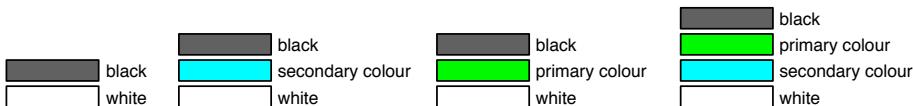


Fig. 2. Four sample habitat occupation patterns ranging between white (the empty grid cell) and black are shown. With three bits there are thirteen different habitat patterns that may arise where an agent must occupy one or more bits of the remaining habitat.

2.3 Agent Reproduction and Death

Agents do not move during the simulation. A few white-inhabiting but otherwise randomly engineered agents are scattered onto the grid at the start of a run. To avoid the need for conflict resolution, a sample of random locations on the board, and random habitat levels within those cells, are selected for update at a pre-specified rate per simulation time step. If an agent occupies the selected cell and level, this agent is updated by one time step. Otherwise this update attempt has no effect on any agent.

In addition to habitat bits, an agent genotype contains an integer *lifespan* measured in agent updates (not simulation time steps). After a fixed proportion of their lifespan agents start to reproduce asexually. Every reproductively mature agent selected for update generates a single child. Children inherit the genotype of their parents with the possibility of a mutation that completely regenerates the selected gene. If the gene selected for mutation is a habit preference or habitat provision gene, the other habitat genes in the genotype are adjusted if necessary to maintain genotype coherence — i.e. so that the habitat provision genes contain a subset of the bits in the habitat required genes.

A potential parent randomly samples cells in its Moore neighbourhood (including its own cell), searching for suitable habitat for its offspring. If none is located in this sampling, the child dies immediately and does not appear on the grid. Otherwise, the parent deposits the offspring in the empty white cell or on top of any pre-existing agents that provide suitable higher-level habitat. Thus it is possible for a child to occupy the habitat provided by its parent if a fortuitous mutation permits this.

An agent that reaches its lifespan is removed immediately from the grid. Any agent that depended on the dead agent for habitat is also extinguished up the habitat chain of that cell. Thus if a base-level agent dies, all agents on its cell die simultaneously for want of suitable habitat and the white base level habitat will become accessible again.

3 Results

The software is coded in C++ employing the OpenGL and OpenAL APIs on an Apple Macintosh, running OSX. The basic parameters used to establish the results described here are as follows.

- Grids from 10x10 to 50x50 have been tested. Grid size doesn't make a significant impact on the behaviour of the simulation although as the resolution is reduced obviously the scope for pattern formation decreases.
- The grid is initialised with a fixed white (1,1,1) base habitat.
- The grid is initialised with 5 random, white-living agents regardless of its size.
- An agent's reproductive maturity is fixed at 80% of its lifespan.
- The maximum life span of an agent is 100 updates (lifespan is evolvable).
- 1 randomly selected level on 50% of cells is selected for update at each simulation time step. If a selected cell and level is uninhabited, nothing occurs.
- The mutation rate is set to 1/100.
- Would-be parents make 4 attempts to find local habitat for their offspring.

The simulation parameters are few in order to examine, in isolation, the potential of habitat creation as a generative mechanism of sonic and visual complexity. Hence there are many aspects of organism reproduction, movement, niche construction and ecosystem engineering that could be considered. The potential for future work is discussed below.

3.1 Visualisation and Basic Simulation Results

Figure 3 shows a typical *Habitat* visual sequence showing the rich, patchy landscape that emerges. The structure of the landscape remains spatiotemporally stable, even across large time frames, however it is anything but static. For instance, note the existence of a stable magenta habitat towards the upper-left of the screenshots. The boundary of this formation changes but the structure remains over the 20,000 time steps. Note the development of a stretch of green habitat on the lower right hand side of the images that coalesces from three discontinuous patches.

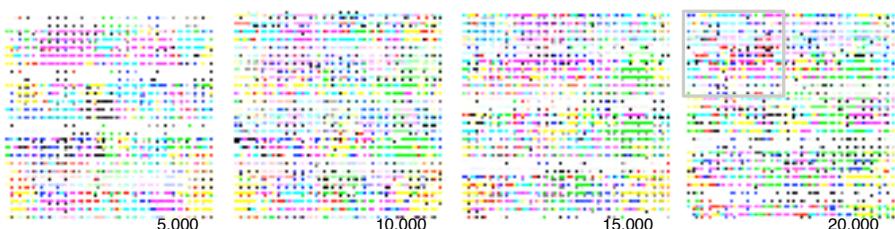


Fig. 3. Images of *Habitat* at 5000 frame intervals. The faint grey rectangle (*top left at 20,000 frames*) indicates the area enlarged in Figure 4.

Figure 4 shows a close up of the simulation at 20,000 frames. The width of the bars drawn in each cell represents the lifespan of the agent that is providing the habitat of the indicated colour at that location. A full cell-width indicates the maximum life span of 100 updates.

As the stack height increases, agents of decreasing lifespan form stable cells. A short-lived agent cannot provide a basis upon which others can depend for habitat. Reciprocally, a long-lived agent cannot live on a shorter-lived habitat provider and hope to achieve reproductive maturity. Hence, miniature *Towers of Hanoi* emerge naturally from the simulation and are only broken by a mutation that produces an unsustainable top-heavy configuration.

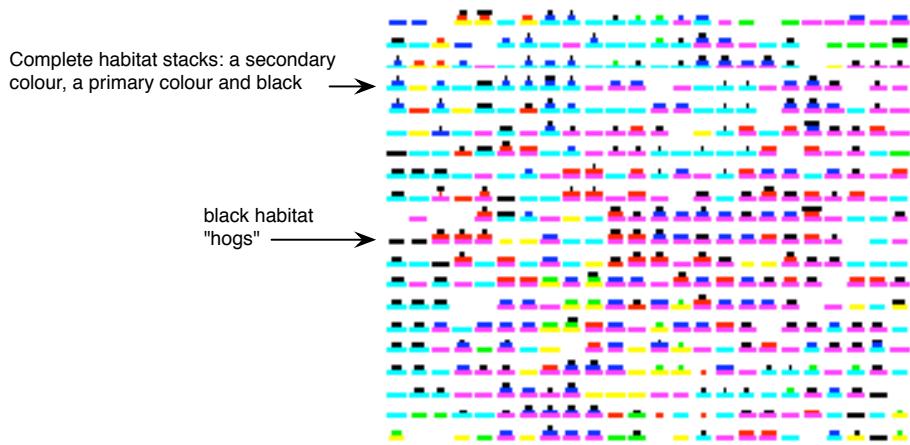


Fig. 4. Close-up (*top left corner of frame 20,000, Figure 3*) showing individual habitat layers. Agent lifespan is represented by the width of the habitat bar at each location and ranges from 0 to 100 agent updates.

Over hundreds of thousands of time steps the simulation maintains stable habitat configurations. Figure 4 shows some black habitat “hogs” that occupy all bits of their white habitat. These agents appear to be at a selective disadvantage when compared to coloured agents. This is surprising given that black is barren wasteland for other agents and provides no habitat for others to infiltrate. The reasons for this behaviour are yet to be established statistically. However, the phenomenon has been confirmed by seeding *Habitat* exclusively with randomly placed white-living, black-providing agents.

One possible reason for the behaviour is that every reproducing agent samples its neighbourhood, including its own location, for a place to locate its offspring. When the local neighbourhood is densely populated with creatures of a parent’s kind, a mutation will allow the offspring to survive by shifting it to the next upward layer of local habitat. This is most likely identical to the habitat provided by the parent agent itself. Unfortunately for black agents, no children, regardless of any mutation they possess, can live on its parent or its parent’s kind. Thus black wastelands appear to be colonized from the edges by fitter agents that have the option of placing offspring in habitat provided by their own kind (when fortuitous mutations arise), and then opportunistically mutating back to their parent’s kind when conditions allow.

3.2 Sonification Results

Habitat simulates evolutionary time periods. Our software therefore allows for the sonification of the evolutionary dynamics of habitat formation. Alternatively, a set of habitats that have evolved over a period of time may be evolutionarily frozen by the user and sonified directly.

To generate a spatialised, coherent, evolutionary sonic ecosystem from *Habitat*, each agent type is assigned audio characteristic of a real organism according to the type of habitat that it requires, and the type that it provides. These are stored as sampled audio files for playback. Agents that are occupying the base-level habitat are assumed to be autotrophs and are assigned the sound of marsh grass, bushes or trees blowing in a gentle breeze. Agents that occupy the grass, bushes or trees are assigned samples of distinct animal calls found in the wetland environment located down the street from the author's home. In the prototype described here, the plants are occupied by the Common Eastern Froglet (*Crinia signifera*), the expressively named Eastern Pobblebonk (*Limnodynastes dumerilii*) and the Spotted Marsh Frog (*Limnodynastes tasmaniensis*). Finally, black habitat is assigned the percussive chirp of a cicada, the Alarm Clock Squeaker (*Pauropsalta mneme*)¹. Agents generate audio only when they are reproductively mature. See section 4 for a description of the intended final soundscape.

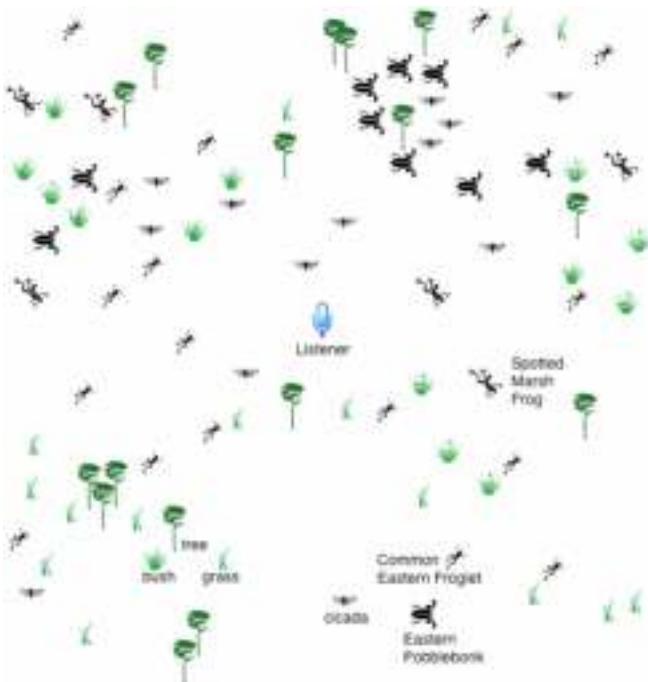


Fig. 5. An illustrative, cumulative section of the score produced by sonifications over 60 seconds with evolution paused at 10,000 time steps

¹ Real cicadas don't depend on frogs or birds for habitat production, they live in trees! This prototype was based largely on aesthetic grounds. See section 4 for a more easily justified implementation.

The listener is positioned in the centre of the grid (Figure 5). The sound of the environment is spatialised around the listener according to the direction of the generating sources and attenuated according to the generator's distance from the listener. To avoid a cacophony of mature singing agents, the grid is sampled sparsely at each frame for audio generation. The current sampling density is 1 sample per frame with the simulation running at 25 frames per second. This can be altered to vary the density of sounds generated by the software. If a mature agent is located at the sampled position and habitat level, that agent voices its audio file.

Figure 5 shows a cumulative "score" generated over a one-minute period after evolution has been frozen at frame 10,000. By freezing the evolutionary process and preventing the agents from ageing, we maintain a snapshot of the habitat as it has evolved to this point. This is suitable for generating a coherent audible experience from a single habitat arrangement. Alternatively, evolution may be left running and the software will generate an evolutionary soundscape.

The resulting soundscape is most pleasing to the author when the grid contains about twenty by twenty cells. This gives sufficient density without overpowering the listener with too many individual sound sources. The number of organisms, in particular the number of frogs, seems to approximate the effect of the author's local wetland and the repeated calls made by particular frogs is regular enough to give the desired effect of a spatially coherent handful of callers. Although some ponds are literally hopping with singing frogs it has not been the aim to replicate this din!

4 Future Work

This simulation and visualisation could be extended in many ways. A few that are relevant to constructing a coherent habitat model and a soundscape are listed below.

An extension to the application in generative art is the inclusion of multiple habitats in a single layer within a grid cell. A white cell could provide habitat for three agents simultaneously and at the same level if they were red, blue and green, or perhaps for two agents simultaneously at the same level if they were magenta and green. The magenta agent could provide habitat for a red and a blue agent on its own back. This would more accurately reflect habitat formation by ecosystem engineers such as the tree that simultaneously provides different kinds of habitat for occupation by beneficiary species. For instance, a particular tree may harbour earwigs under its bark and possums in its hollows. The software would benefit from a longer habitat bit pattern to allow for this greater species diversity.

With a longer habitat bit pattern, it is possible to diversify the sonic material played by the software for installation in a multi-speaker environment. As well as the frogs listed above, endemic thicket-hiding and tree-dwelling birds will be added. The Noisy Miner, Wattlebird and Magpie are suitable candidates. Cicadas will be assigned specifically as tree dwelling organisms.

To sonify this intricate environment a "hierarchical listening" approach would be helpful. Rather than sonify the whole environment simultaneously (as occurs now), the listener could focus attention at different scales, from close-up to wide overview, and also move around within the grid world. Agents in a neighbourhood would then play a chorus dependent on the species present in the vicinity of the listener. In this way a walk through *Habitat* could be simulated.

It is possible to directly seed the simulation base-habitat cells with colour to structurally predetermine habitats. Predetermined habitats give specific species an advantage in some areas and not in others. For instance a cyan region could be drawn into the base-grid ensuring that no secondary coloured agents could ever inhabit this location — just as sand or water might prevent bushes and trees taking hold. Pre-established habitats like this would change the dynamics of the simulation and sonification considerably. They would act as simulated ponds or arid areas with their own characteristic sounds.

A continuous model of habitat occupation would permit agents to survive in less than ideal areas in times of hardship, even if they were not able to grow or reproduce there. This would be particularly useful if the software was applied to ecology.

5 Conclusion

This paper presents a simple simulation of an evolutionary ecosystem from which emerges stable, multi-layered habitats based on ecosystem-engineer agents. The habitats form spatiotemporally coherent, patchy environments. They are generated by asexually reproducing, immobile agents operating on a two dimensional grid. Each agent engineers a new form of habitat suited to occupation by other agents. Agent lifespan evolves within the simulation to create stacks of habitat with long-lived agents at their bases, and progressively shorter-lived agents on top.

The simulation has been used to generate a soundscape that represents the experience of a Melbournian urban wetland. It can also be operated in a mode that sonifies the simulated evolutionary process as it unfolds.

Acknowledgements

Thanks Ollie Bown, Alice Eldridge, Volker Grimm, Kevin Korb, Jon McCormack, Peter McIlwain, Ben Porter, Suzanne Sadedin and the reviewers for recent discussions and correspondence that assisted in formulating the ideas presented here. This work was funded by Australian Research Council discovery project grant DP0772667.

References

1. Berry, R., et al.: Unfinished Symphonies - Songs of 3.5 worlds. In: Workshop on Artificial Life Models for Musical Applications, 6th Euro. Conf. on Artificial Life, pp. 51–64. Editriale Bios, Prague (2001)
2. Dahlstedt, P.: Living Melodies: Coevolution of Sonic Communication. In: First Iteration, pp. 56–66. CEMA, Melbourne (1999)
3. Dorin, A.: The Virtual Ecosystem as Generative Electronic Art. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 467–476. Springer, Heidelberg (2004)
4. McCormack, J.: Eden: An evolutionary sonic ecosystem. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS, vol. 2159, pp. 133–142. Springer, Heidelberg (2001)

5. Dorin, A., Korb, K.: Building Artificial Ecosystems from Artificial Chemistry. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS, vol. 4648, pp. 103–112. Springer, Heidelberg (2007)
6. Jones, C.G., Lawton, J.H., Shachak, M.: Positive and negative effects of organisms as physical ecosystem engineers. *Ecology* 78(7), 1946–1957 (1997)
7. Odling-Smee, F.J., Laland, K.N., Feldman, M.W.: Niche Construction, the neglected process in evolution. In: Levin, S.A., Horn, H.S. (eds.) Monographs in Population Biology. Princeton University Press, Princeton (2003)
8. Whitelaw, M.: Metacreation: Art and Artificial Life. MIT Press, Cambridge (2004)
9. Miranda, E.R.: On the Evolution of Music in a Society of Self-Taught Digital Creatures. *Digital Creativity* 14(1), 29–42 (2003)
10. McCormack, J.: Aesthetic Evolution of L-Systems Revisited. In: Raidl, G.R., et al. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 477–488. Springer, Heidelberg (2004)
11. Greenfield, G.: On Evolving Multi-Pheromone Ant Paintings. In: IEEE Congress on Evo. Comp., CEC, Vancouver, pp. 2072–2078. IEEE, Los Alamitos (2006),
<http://ieeexplore.ieee.org>
12. Bentley, P.J., Corne, D.W.: Creative Evolutionary Systems. Morgan Kaufmann, San Diego (2002)
13. Dorin, A.: Artificial life, death and epidemics in evolutionary, generative electronic art. In: Rothlauf, F., et al. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 448–457. Springer, Heidelberg (2005)

The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine

Arne Eigenfeldt

School for the Contemporary Arts, Simon Fraser University
Burnaby, Canada
arne_e@sfu.ca

Abstract. This paper presents an evolutionary music system that generates complex rhythmic polyphony in performance. A population of rhythms is derived from analysis of source material, using a first order Markov chain derived from subdivision transitions. The population evolves in performance, and each generation is analysed to provide rules for subsequent generations.

Keywords: Rhythm generation, genetic algorithms, recombination.

1 Introduction

Kinetic Engine [1, 2, 3] is an evolving interactive musical performance software system that generates complex rhythmic polyphony using musically intelligent algorithms. In its three incarnations to date, it has been used as an installation, a networked performance using ten computers, as an instrument within an ensemble of improvising musicians, an intelligent controller for a 12-armed musical percussion robot, and as a composer of a score for percussion quartet. Addressing some of the limitations in earlier versions, version 3 (KE3) uses an evolving population of rhythms produced using a genetic algorithm derived from analysis of source material.

Genetic algorithms (GA) have been used in a variety of ways to create music, both in performance (realtime) and in the studio. Biles, in his discussion of GenJam [4], points out many of the idiosyncratic factors of using GAs in a musical context, including the notion that any fitness test derives from aesthetic judgment, which is inherently difficult to encode [5]. Martins and Miranda suggest that artists use such techniques not as “tools to generate efficient solutions to problems automatically”, but instead, “composers need tools to explore a vast space of possible outcomes” [6]. They also point out that “generating music from algorithms that were not designed for music seldom produces significant pieces of music”, and instead, composers should modify A-Life algorithms to address musical issues [6]. KE3 is one such system, in which a modified GA is employed for musical means: the goal is not any single solution, but the continual evolution of a given data set that is heard as it evolves.

Rather than beginning from a random population, KE3 begins with an initial population derived through analysis of source material. A separate program module is run to extract relevant musical information from user provided MIDI files. This material can be very explicit (i.e. a set of patterns to use for a section of music) or general (i.e.

a transcription of an entire piece of music). The Analysis module parses the material for various aspects, which are passed to multiagent Player modules as XML files. The Player agents generate initial populations that closely resemble the rhythmic aspects of the source material. Individuals within the population are chosen by various means for performance. Evolution occurs in isolation within each agent, and in realtime, during performance. Agents keep track of those individuals that are chosen for performance; heard individuals become more likely to be culled. Since all individuals are deemed acceptable, the system does not choose individuals that are allowed to remain alive, but instead chooses individuals to cull. After each generation has been bred, the new population is analysed, and this new analysis becomes the rule-set for generating new members of the population.

Although diversity depends upon the source material (the basis of the original population), it is further maintained through two mutation methods: the substitution of null rhythms (rests) at the beginning or end of the individual, and the substitution of related chromosomes (rhythmic subdivisions) within the individual. The amount of mutation is under performance control.

Section 2 presents background information, including a brief description of the context of interactive computer music, the difference between realtime composition and improvisation, and earlier versions of Kinetic Engine; Section 3 describes previous research in the field; Section 4 provides a detailed description of the latest version of Kinetic Engine and its relationship to previous systems; Section 5 suggests future directions; Section 6 presents some conclusions.

2 Background

2.1 Interactive Computer Music

Realtime computer music, in which the computer makes compositional decisions in performance that react to composer/performer interactions, has tended to fall within the domain of improvisatory systems¹. In an effort to model the “unpredictability” of improvisation, constrained random procedures have been incorporated so that the musical surface can be both varied, yet easily controlled [7]; for example, choosing pitches randomly from a user determined scale. High level changes require more deterministic programming, or greater user control; for example, changing scales, chords, rhythmic units, etc. Formal cohesion, or large-scale structural logic offered by recapitulation and restatement, remains with the composer/performer.

2.2 Realtime Composition versus Improvisation

With the use of computers, the ability to compose during performance becomes conceivable. Realtime composition [8] is distinct from improvisation in a number of ways, but one chief distinction is musical memory. KE3 separates these two creative methods by incorporating predefined musical structures as source data (composition), as well as allowing spontaneous control during performance, which does not affect future actions (improvisation). In practical terms, composed elements generate the populations, while improvised elements influence how the populations are explored.

¹ See [15] for an overview of early interactive systems and methodologies.

2.3 Previous Systems

Kinetic Engine began as an effort to place more high-level compositional responsibility within software. In attempting to incorporate aspects of artificial intelligence into music performance software, the decision was made to limit the exploration to rhythmic development and interaction.

2.3.1 Kinetic Engine V.1

Kinetic Engine version 1 (KE1) focused on the interrelationship of simple parts to create continuously varying rhythmic interplay between four virtual players. Basic rules were defined as to how players generated individual parts; more complex rules were defined as to how these parts interacted.

2.3.2 Kinetic Engine V.2

Kinetic Engine version 2 introduced performative control, and furthered the notion of intelligent interaction between individual players through autonomous multiagents. Agents generated rhythms in response to a changing environment. Once these rhythms had been generated, agents “listened” to one another, and altered their patterns based upon these relationships. Since the rule-set for rhythm generation remained static, the resulting music, while initially engaging, remained essentially homogeneous. Furthermore, since the complex interactions were based upon seeded probabilities, the music was essentially improvisatory and afforded limited compositional control.

2.4 Kinetic Engine V.3

The latest version of Kinetic Engine attempts to balance spontaneous change with the ability to alter and adapt its rule-set. As such, KE3 offers the potential for both real-time composition through recombinance [9] as well as improvisation through generative means. Section 4 describes the software in detail.

3 Previous Research

One of the first realtime music applications of genetic algorithms, *GenJam* [4] performed jazz improvisations by creating a population of melodic phrases which had been evolved in one of two ways: under the interactive guidance of a human mentor, or by eliminating the fitness bottleneck entirely by populating the database with pre-selected data.

More recently, Martins and Miranda [10] created an A-Life system in which a collective rhythmic database evolved among a group of musical agents. They state at the outset that the system was based upon language imitation games; as such, it was more experimental and conceptual, rather than a system that produced successful music. The initial database of rhythms was randomly generated, and individuals were passed between agents in order to evolve a repertoire.

Alfonseca et al. [11] used a genetic algorithm to generate music in a pre-defined style. GAs have proven to be useful in these situations, since the goal (stylistic emulation) is objective, and thus a fitness function can be more clearly defined. Similarly,

Mararis et al. [12] created artificial music critics, trained via a neural net using 992 musical works, that evaluated the artifacts created by an evolutionary music composer: the goal of the evolutionary composer was to create “aesthetically pleasing” variations on Bach’s *Invention #13*. Significant correlations were found between the artificial critics and 23 human subjects. As with Alfonseca, it is easier to test a system’s success given such objective criteria: personal systems, whose goal is to create new music, are more difficult to judge, since their success depends upon the ability to recreate the composer’s aesthetic, one which may not be judged as “aesthetically pleasing” by listeners. This, according to Nic Collins, is “playing the composer card” (personal communication).

More relevant to the research presented here is the work of Weinberg et al. [13], who used a GA to create a population of melodies used by a robotic performer in response to a human improviser. A number of melodic excerpts of variable lengths and styles were transcribed from an improvising pianist, and served as an initial population. Individuals were chosen in realtime in response to input melodies. The fitness test measured similarity to the input phrase.

Waschka’s *GenDash* [14] employs evolutionary computation algorithms to “help compose” musical works. Similarities to KE3 include an initial population chosen by the user, and a stochastic fitness function. *GenDash* is not a realtime system.

4 Kinetic Engine Version 3

KE3 is unique in several ways. First, although its population is comprised of rhythmic representations, its evolution occurs via transitions found within the population, rather than altering the population directly. Second, transition data found in individuals - used to generate successive generations - is not accumulated into a single table, but remains discrete. This avoids a type of neutralizing, or averaging, of the data, and results in closer variations to individual rhythmic phrases. Third, unlike many machine learning systems, KE3 does not attempt to learn general rules from a large dataset; instead, the author recognises that successful music is entirely context-dependent. As such, specific examples are given to the system in order to deduce rules specific to a desired section of music: these rules are immediately forgotten.

The use of GAs within composition, even in realtime, is well established; even the avoidance of a using a fitness function has been implemented previously. KE3’s initial design did not include a GA: its inclusion literally evolved during development from artistic needs. Separating analysis from performance allowed for the creation of a potential pool of available material for Player agents - the musical development of this material naturally suggested GA and its unique features.

Furthermore, the use of a GA permitted the concept of musical memory to enter the system: rhythms do not emerge and disappear (as they do with most improvisatory systems, including KE2); instead, they evolve, while potentially retaining older versions. The ability to resurrect older individuals by re-introducing them into the population is a further example of exploiting this concept.

Almost all of the design decisions have an artistic motivation, including the avoidance of the fitness test. Although a fitness test is not applied before reproduction, evaluation functions definitely remain. The fitness of any individual will be its

suitability to the current musical environment: since the environment is constantly changing, the suitability of any individual will likewise change. Therefore, those individuals that meet the immediate requirements are selected for performance (see section 4.6). Furthermore, fitness will change over time through repetition: musical composition is a balance between repetition and variation, familiarity and novelty. The more an individual is selected for performance (and is heard), the less useful it becomes. However, continual variation through the introduction of new data is also not ideal, since some degree of repetition is required; thus, some older individuals need to survive in order to maintain musical cohesion².

4.1 Design

See figure 1 for an outline of the design of KE3.

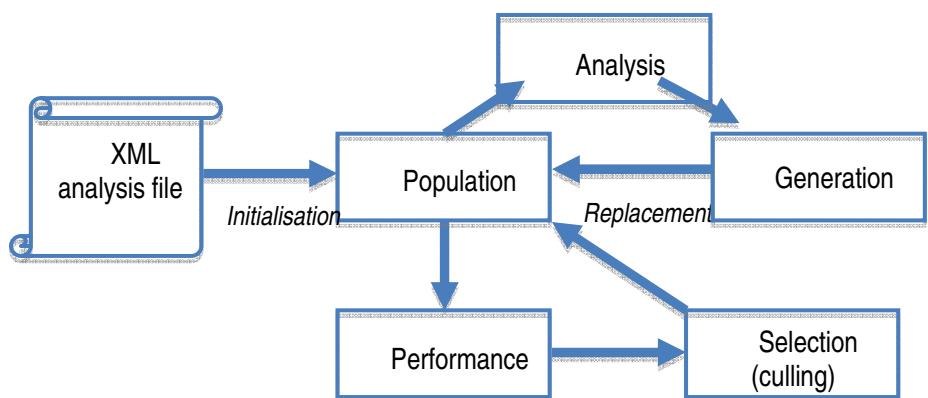


Fig. 1. The design scheme for Kinetic Engine version 3

4.2 Representation

Kinetic Engine uses a chromosome structure for representing rhythmic phrases of indices into a database of potential rhythmic subdivisions of a beat. See figure 2 for the first sixteen representations.



Fig. 2. The first sixteen subdivisions within the subdivision array

² Clearly, these statements are reflections of the author's compositional aesthetic: readers may differ in their own musical values.

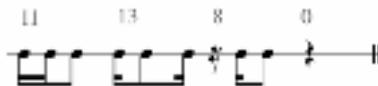


Fig. 3. Example rhythm and its representation

Thus, the following rhythm is represented internally within Kinetic Engine as (11 13 8 0):

Individuals can be of varying length, based upon the phrase lengths detected by the analysis module. Pitches data will not be discussed here.

4.3 Initialisation and Population Analysis

Loading an XML analysis file constitutes initialisation. As well as providing the initial population, data from this file includes a similarity analysis between phrases (used by the Player to determine how *much* variation will occur through selection) and sectional periods (used to determine how *often* to make variations).

Every new population is immediately analysed after generation for rules on how to breed the next population. Transitions between subdivision representations are generated using Markov analysis, and stored in a multi-dimensional matrix in order to separate individual beat densities. Population analysis also includes an individual's density (the number of events per rhythmic pattern compared to the maximum possible), and complexity (the degree of syncopation within the individual beats and placement within the pattern): both are used by the Player in its selection criteria.

Population size can be controlled in performance, essentially limiting the number of offspring an individual can produce (to a maximum of 32 in total). The Player agent can begin selecting individuals from the population to perform.

Initialisation may take place several times during a performance: whenever new analysis data is loaded into the system. This will result in the complete loss of an existing population, and the generation of an entirely new one. In musical terms, this occurs at each new musical section, where distinct changes are required in material. While this may appear somewhat erratic from an A-Life standpoint, it addresses a major problem with a great deal of realtime computer music, and improvisation in general: the inability to quickly change musical direction (or “turn on a dime” as one musician described it).

4.4 Selection for Elimination

Since all individuals in a population are children of strong solutions, all are considered fit to live (and be heard). Depending upon the musical environment, in most cases only a small percentage of the population will actually be heard. Individuals selected seldom or not at all by the Player remain “fresh”, while those that are selected many times are considered “stale”. The number of times an individual is selected for performance is recorded, and those with higher rankings in this regard have a greater chance of elimination.

The elimination algorithm is a variation of the roulette wheel selection (RWS): the amount of “culling” is user defined (the default is 50%). The population is sorted by usage, and a random value is exponentially scaled to select overused patterns. This method does not guarantee elimination of weak candidates (those that are stale), nor the prolongation of strong candidates (those that remain fresh). Musically, this maintains an unpredictable balance between the repetition of previous material with the introduction of new material.

4.5 Generation and Replacement (Reproduction)

Breeding is user initiated during performance. KE3 does not combine individuals through crossover, as this was found to create unmusical rhythms. Instead, the genetic material of a single parent is used to intelligently create a variation. Special attention is paid to beginnings and endings of rhythmic phrases through independent transition matrices for these points.



Fig. 4. A parent and four children

The result of restricting evolution to single parents is a population expanding along divergent paths, as well as increasing the potential for genetic drift. Furthermore, due to the small population, allele may be entirely lost, thus limiting potential divergence. For this reason, it is possible during breeding to reintroduce ancestors – the original individuals from source data – to the genetic pool. Although the number and exact members selected are stochastically chosen, this essentially restocks the genetic pool, preventing, or at least limiting, both convergence and divergence.

As with all GAs, mutation is necessary to avoid stagnation and allow unpredictable elements to enter the population. KE3 has two basic mutation algorithms. The first technique is a Stravinsky-like substitution of rests for notes at the beginnings and ends of patterns; this ensures that an individual’s length remains constant, as well as the specific placement of subdivisions on intended beats. The second algorithm substitutes similar chromosomes (similarity being determined by an algorithm that compares weighted onsets and density) within the patterns themselves. In both cases, the degree of mutation is constrained during performance: this amounts to a compositional control, as it has lasting effects.



Fig. 5. A chromosome (a), and two similar chromosomes, (b and c) and two dissimilar (d and e). (b) and (c) are thus potential mutation substitutions, whereas (d) and (e) are not. Similarity coefficients are displayed in relation to (a).

4.6 Selection for Performance

KE3 uses a sequencer model for time, in which measures and beats are counted using a standard transport mechanism. The Player agent continually generates an array of future events - considered the agent's Intention - derived from the available rhythms and pitches in its population. Which individuals are chosen depends upon the musical environment: player agents look to the user-set global density and complexity parameters, and use a K-nearest neighbor algorithm to choose a class of individuals from the population that best match these values.

Each agent calculates a variation vector, which determines how often to change individual patterns, and how much to change by. This vector is based upon analysis of the original source material: if the source material is deemed to have a high diversity in terms of density, complexity, and similarity, the agent will assume a similar diversity in choosing individuals from the population to perform. Similarly, the amount of repetition within the source material will determine how often new individuals will be chosen. Variation is, therefore, under compositional control.

4.6.1 Generative Control and Adaptation

In certain situations, individuals in the existing population may not adequately meet the musical requirements set by the user during performance. In these cases, temporary adaptations may be made to the data contained in the agent's Intention by adjusting the complexity (syncopation) and density (number of events per beat) to match the user-set parameters. As these variations occur outside the population, they are not retained within it, and are thus improvisational responses to the environment.

5 Future Directions

Several improvements can be made to Kinetic Engine version 3, some of which have resulted in the beginnings of Kinetic Engine version 4. The ability to accept audio input as source material has already begun, taking into account the subtle variations in timing and volume that can be considered expressive performance. These parameters will be analysed and used by the Player agents.

Somewhat ironically, the most logical extension of Kinetic Engine has been on the "to do" list the longest, that of incorporating pitch and harmonic analysis to create non-percussive: this is the basis of Kinetic Engine version 4. The initial limitation in KE1 on rhythm was for the practical reason of constraining the search domain; however, as new areas of rhythmic complexity and structuring have continued to present themselves, the "limit" upon rhythm has not proven "limiting" at all.

6 Conclusion

The ability of genetic algorithms to produce consistently good solutions is one reason that they have become a useful A-Life technique for computer-assisted composition. However, the necessity of testing the quality of population members has reduced its usefulness in realtime situations. Several researchers have come up with alternatives to the fitness bottleneck, and KE3 uses one method first used by Biles in *GenJam*: pre-populating the database with pre-selected data. However, unlike Biles, the data is derived from analysis, rather than the direct incorporation of source material.

The latest version of Kinetic Engine continues with many of the techniques implemented in its earlier incarnations in an effort to bring musical intelligence to the realtime decision-making. The use of a population of GA-produced rhythms and interrelated pitches as a potential pool of patterns from which an autonomous Player agent can choose is a major addition to this evolving software. Furthermore, it allows for a potential balance between the instantaneous change of improvisation, and the predetermination of composition.

Acknowledgements

This research was undertaken through a Research/Creation grant from Canada's Social Science and Humanities Research Council (SSHRC).

Thanks to Eduardo Miranda, Alexis Kirke, and Torsten Anders for their suggestions and comments during a brief residency at the University of Plymouth's Future Music Lab.

References

1. Eigenfeldt, A.: Kinetic Engine: Toward an Intelligent Improvising Instrument. In: Proceedings of the Sound and Music Computing Conference, Marseille (2006)
2. Eigenfeldt, A.: Drum Circle: Intelligent Agents in Max/MSP. In: Proceedings of the International Computer Music Conference, Copenhagen (2007)
3. Eigenfeldt, A.: Multiagent Modeling of Complex Rhythmic Interactions in Realtime Performance. In: Sounds of Artificial Life: Breeding Music with Digital Biology, A-R Editions (2008)
4. Biles, J.A.: GenJam: A Genetic Algorithm for Generating Jazz Solos. In: Proceedings of the International Computer Music Conference, San Francisco (1994)
5. Biles, J.A.: Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness. In: Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program, San Francisco (2001)
6. Martins, J., Miranda, E.R.: Emergent rhythmic phrases in an A-Life environment. In: Proceedings of ECAL Workshop on Music and Artificial Life, Lisbon (2007)
7. Chadabe, J.: Interactive Composing. Computer Music Journal 8 (1984)
8. Eigenfeldt, A.: Intelligent Realtime Composition (2008),
http://cec.concordia.ca/econtact/10_4/
9. Cope, D.: Experiments in Musical Intelligence. Middleton, A-R Editions (1996)

10. Martins, J., Miranda, E.R.: A connectionist architecture for the evolution of rhythms. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) *EvoWorkshops 2006. LNCS*, vol. 3907, pp. 696–706. Springer, Heidelberg (2006)
11. Alfonseca, M., Cebrian, M., Ortega, A.: A simple genetic algorithm for music generation by means of algorithmic information theory. In: IEEE Congress on Evolutionary Computation, CEC 2007 (2007)
12. Manaris, B., Roos, P., Machado, P., Krehbiel, D., Pellicoro, L., Romero, J.: A Corpus-based Hybrid Approach to Music Analysis and Composition. In: Proceedings of the 22nd Conference on Artificial Intelligence (AAAI 2007), Vancouver (2007)
13. Weinberg, G., Godfrey, M., Rae, A., Rhoads, J.: A Real-Time Genetic Algorithm in Human-Robot Musical Improvisation. In: Computer Music Modeling and Retrieval, Sense of Sounds. Springer, Berlin (2008)
14. Waschka, R.: Composing with Genetic Algorithms: GenDash. In: Evolutionary Computer Music. Springer, London (2007)
15. Eigenfeldt, A.: Realtime Composition or Computer Improvisation - A Composer's Search for Intelligent Tools in Interactive Computer Music (2007), <http://www.emsnetwork.org/>

Filterscape: Energy Recycling in a Creative Ecosystem

Alice Eldridge and Alan Dorin

Centre for Electronic Media Art, Monash University, 3800, Australia
alice@ecila.org, alan.dorin@infotech.monash.edu.au

Abstract. This paper extends previous work in evolutionary ecosystemic approaches to generative art. *Filterscape*, adopts the implicit fitness specification that is fundamental to this approach and explores the use of resource recycling as a means of generating coherent sonic diversity in a generative sound work. Filterscape agents consume and deposit energy that is manifest in the simulation as sound. Resource recycling is shown to support cooperative as well as competitive survival strategies. In the context of our simulation, these strategies are recognised by their characteristic audible signatures. The model provides a novel means to generate sonic diversity through de-centralised agent interactions.

1 Introduction

In recent years the *ecosystem* has appeared as the metaphorical cornerstone in the work of a number of electro-acoustic composers [1], performers [2] and performance theorists [3]. The unifying theme across these diverse works is that a musical performance is contingent upon a range of interacting components (social, ethnographic, musical, technological etc.) and emerges at performance time, rather than being explicitly determined in advance.

Within the field of Evolutionary Music and Art (EMA), there is a similarly burgeoning interest in ecosystemic concepts. In this field, ‘ecosystem’ refers to a set of systemic principles that extend the metaphors of standard Evolutionary Computation (EC): broadening the metaphor from Darwinian evolution of isolated genotypes to the digital specification of entire ecosystems may be a fruitful approach in creative contexts [4], [5]. In this case ‘fitness’ is no longer an explicitly pre-specified function as in standard EC, but implicit in the design of agents’ interaction with their environment and one another. In parallel with broader musical trends, the resultant sonic behaviour is contingent upon the interacting components and emerges at runtime.

For the artist wishing to explore the potential of ecosystemic models, a generalisable and comprehensive body of work, similar to that which currently exists in the EC literature, has yet to be formulated. In an ongoing project, we are exploring the application of specific ecosystemic principles in audio-visual contexts (e.g. [6], [7]) with the aim of developing a conceptual and methodological tool box for EMA practitioners.

This paper addresses the issue of evolutionary ‘selection pressure’ and aesthetic diversity. A study in the design of a sonic ecosystem is presented in which energy recycling is shown to generate novel survival strategies, leading to an increase in the range of sonic behaviour that is generated.

1.1 From Fitness Functions to Survival Strategies

In moving from standard EC to an evolutionary ecosystemic approach the isolated genotype is embedded in an environment. An agent’s chances of survival and reproduction are implicitly determined by its environmental interactions, rather than by an explicitly defined fitness function. This reflects a theoretical shift from adaptationist Darwinian evolution to extended evolution [8]. Adoption of this stance is fuelled partly by the belief that it is a more accurate reflection of the actual processes of natural evolution, and partly the intuition that this may be a more natural – and ultimately rewarding – approach to EMA.

From this ecosystemic perspective, selection pressures emerge through the evolution of agents mediated by their environment. The specification of agent-environment interactions replaces the fitness function as the principle design challenge. This task can be informed by existing Artificial Life (Alife) models coupled with insights from ecology. For example, in many existing agent-based models, reproductive success is determined by individuals’ energy acquisition ability rather than a global fitness function.

Existing models demonstrate that heterogeneity of resources in the environment can give rise to surprisingly complex agent behaviour at the population level [9]. Our intuition is that if interesting dynamics can emerge from agent-environment interactions with fixed environments, then modelling situations in which agents can *alter* their environment has great potential in creative contexts, particularly in situations where coherent diversity is desirable. The application of this process of *niche construction* in EMA is currently under investigation.

The maintenance of diversity is central in many evolutionary music situations. Diachronic diversity can be successfully sustained through the use of coevolutionary designs that ameliorate the problem of premature fitness function convergence. In musical applications this is invariably framed as the competitive coevolution of ‘singers’ and ‘listeners’ (e.g. [10], [11]). Organism interactions are not limited to direct competition however: the myriad of symbiotic relationships observed in the natural world provide rich inspiration for achieving synchronic diversity in creative system design. The study reported here aims to increase the range of potential sonic dynamics by expanding the types of viable survival strategies. Specifically, the standard energy model deployed in agent based simulations is developed to include a simple form of energy recycling with the objective of supporting cooperative as well as competitive survival strategies.

1.2 Energy Recycling and Cooperative Behaviour

In models such as the early versions of *Sugarscape*, resources are externally supplied and agents compete to collect them by traversing a simulated landscape. Such models capture the basic flow of energy from *producers* to *consumers* in

natural ecosystems. Producers are autotrophic organisms, such as green plants, that are able to manufacture food from the abiotic environment. Consumers are heterotrophic organisms, such as herbivores and carnivores, that rely on eating other organisms to survive. In systems such as *Tierra* [12] we see a slightly more complex food chain with the emergence of parasites which are secondary consumers - agents that exploit other agents to obtain energy.

The flow of energy from producers up to consumers through the trophic levels that is modeled in these systems is known as the *grazer system* in ecology [13]. In real-world food webs the grazer system is invariably coupled with a *decomposer system*. Decomposers (chiefly bacteria and fungi) act to *recycle* nutrients and energy from the dead organic matter produced by the organisms of the grazer system. This recycling is absolutely fundamental to ecosystem functioning but is invariably overlooked in evolutionary simulations [14]. Expanding the basic energy model to include recycling opens the door for mutualistic as well as competitive survival strategies as agents can potentially survive by reusing each other's waste as well as competing for resources.

The remainder of this paper describes the development and analysis of a sonic ecosystem that incorporates a simple form of energy recycling. The system illustrates how the design of low level interactions can give rise to a range of creatively pertinent survival strategies that include cooperative as well as competitive relations. Analysis of the system reveals how environmental structure can be manipulated to influence the prevalence of these strategies. Each survival strategy creates a sonically distinct behaviour. Increasing the repertoire of strategies therefore opens up the range of sonic possibilities.

2 Filterscape

Filterscape is a simple sonic ecosystem in which agents traverse a spectral filter, extracting energy from one frequency band and re-depositing it in another. Sonically, the experiment can be seen as a study in the design of a self-steering spectral filter. Aesthetically, the only constraint is that it should exhibit a coherent and varied range of behavioural dynamics. Conceptually, the study demonstrates how multiple survival strategies can emerge using simple ecosystemic principles. Technically, the aim was to investigate whether these strategies could be shaped by altering environmental structure.

2.1 Model Details

The filter is driven by the movement of an evolving population of agents across its frequency bands. The simulated environment is therefore a discrete, bounded 1D line. The whole population is updated asynchronously at every time step.

Energy Model. Resources are supplied externally. Uniform and non-uniform distribution regimes were compared. In the uniform case, resources are evenly distributed across the space. In the non-uniform case, the total incoming resource

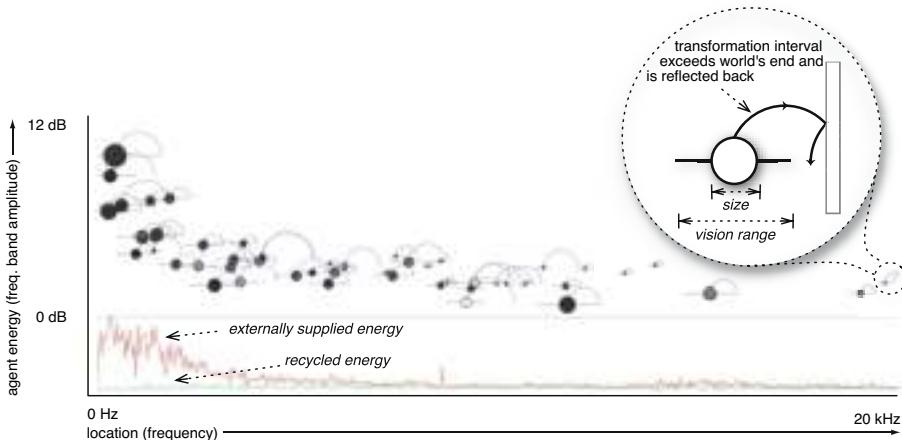


Fig. 1. Screen shot of a Filterscape visualisation. Circle diameter denotes agent size, the arc represents the transformation interval. The horizontal line through the body of each agents signifies its vision range. This image depicts the population shortly after a population explosion. Inset shows how recycled energy is reflected at boundaries.

amount is randomly partitioned and each partition allocated to a location in the world using a uniform random distribution. Resources are added at the same constant rate per time step in both distribution models and unused resources accrue. Energy in each location is equally available to all agents at that location. Rate of incoming energy is set to 35 units per time step.

Agent Specification. Agents are genetically specified by their size and *transformation interval*. An agent's size determines its vision range and metabolism, creating a trade off between energetic requirements and resource acquisition capabilities. The transformation interval determines the location of recycled energy relative to the agent's position (magnitude and direction). Each agent keeps track of its current state as described by its energy level and position. At the start of a simulation run, agents are initialised with $E = 100$ initial energy units and randomly assigned size and transformation intervals.

Agent Behaviour. Agents survive by navigating to and consuming resources. Resources are transformed directly into agent energy units. At each time step, each agent moves to the location with the highest resource level within its vision range. The agent acquires half the resources at its new location. Multiple agents can occupy the same location. An agent dies if its energy level falls to zero.

Movement has an energetic cost proportional to distance moved. Irrespective of movement, a metabolic living cost is imposed as a function of size. At each time step, a percentage of this waste energy is reintroduced into the environment at a location specified by the agent's genetically determined transformation interval. Recycled energy is not differentiated from the external energy supply and can be used in exactly the same way. When transformations would exceed the bounds

of the world, energy is reflected back (see Figure 1, inset). Metabolic living cost is set at 1 unit per unit size; 95% of metabolised energy is recycled.

Reproduction. Haploid asexual reproduction occurs for any agent when its energy level exceeds its initial endowment (E) by a fixed percentage. The offspring produced acquires half the parent's energy. Size and interval are creep mutated with a fixed probability, using values drawn from a Gaussian distribution and scaled over the range of allowable values. The offspring is placed in the nearest empty location. A carrying capacity was externally imposed to prevent population explosions in this simple model, meaning that population fluctuations can occur up to a given limit. Reproduction threshold, $thresh_r = 0.5 \times E$; $P(\text{mutation}) = 0.1$; carrying capacity = 40.

Simulation Parameter Values. Parameter values were set experimentally such that a balance between saturation (agents do not need to move) and starvation (agents are unable to exist) was achieved, whilst maintaining a modicum of biological plausibility.

Sonic Interpretation. The combined energy levels of all agents at a particular location specify the amplitude of the corresponding frequency band. Sound is produced by convolving the array of energy values at each location with white noise and performing an inverse Fast Fourier Transform. Agents therefore act to shift energy from the spectral bin where they are located either up or down the frequency spectrum according to their genetic makeup.

3 Results

3.1 Behavioural Strategies and Sonic Dynamics of the Basic Model

With energy recycling turned on, at least three distinct strategies for resource acquisition can be observed, each generating characteristic sonic dynamics. Firstly, agents can *glide* slowly across the space. Secondly in this bounded world agents can *loiter* at an edge and reflect the energy that they recycle into locations near or within their vision range. Finally agents can exploit each other's recycling behaviour and *cluster* together in mutualistic groups, each taking up the waste deposited by other agents (see Figure 2).

- **Gliding** produces continuous wide frequency sweeps as agents move up and down the filter. This is only a viable strategy in relatively empty worlds where the resource supply is effectively continuous in space. In general they move in the direction specified by their transformation interval collecting their recently recycled energy.
- **Loitering** creates a quiet drone followed by either silence, or a large sweep across frequencies. This is an efficient strategy as agents can reflect energy back within their vision range, minimising the cost of movement. Loiterers make small movements in either direction whilst remaining close to an edge. Often loiterers are the last remaining agents in a population and are invariably precursive to a population extinction or explosion (see below).

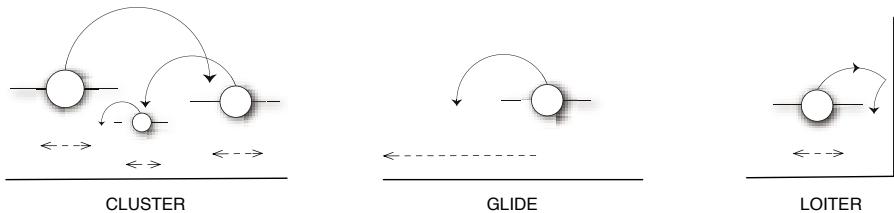


Fig. 2. Agent formations and movement patterns for the principle survival strategies

- **Clustering** creates spectral clusters that drift through frequency space. Clusters can only form when a number of agents' transformation intervals are matched. When this does occur, it offers an efficient strategy as agents' movement costs are minimised.

To confirm that the observed strategies were a product of recycling rather than an artefact of some other aspect of the simulation, recycling was turned off and the external energy rate adjusted. In the absence of recycling, neither clustering nor loitering are observed, leaving gliding as the only viable survival strategy. Recycling supports the viability of survival strategies that increase the repertoire of the system as a sound work. The observation that clustering and loitering are unstable and require additional externally supplied energy led to the hypothesis that their viability is a function of resource distribution.

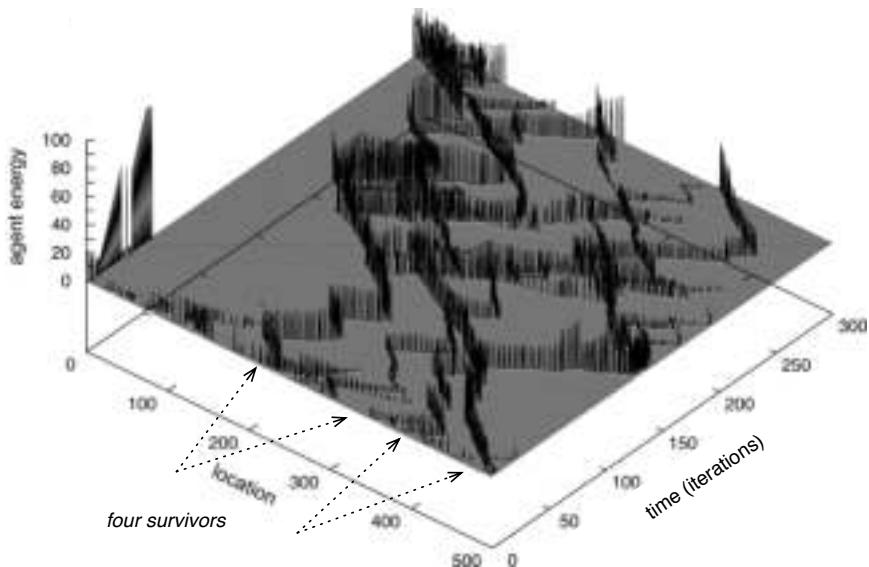
3.2 Effect of Resource Distribution on Survival Strategy and Population Dynamics

To examine whether the prevalence of the different behavioural strategies (glide, loiter, cluster) were influenced by resource distribution, the simulation was run using uniform and non-uniform resource allocation (see section 2.1).

Figure 3 shows typical plots of resource deposits made by agents under uniform (top) and non-uniform (bottom) distribution regimes. Each agent deposits energy at every timestep, therefore these give an imprint of the movement of the population in space-time. Figure 3 (top) illustrates the typical behaviour of a population with uniformly distributed resources: individuals glide smoothly back and forth through space. In contrast Figure 3 (bottom) shows a space-time plot for the loitering behaviour that is most prevalent in the non-uniform condition. A small number of loiterers hug the edge of the space from iteration 100 until around 170 when a population explosion occurs. The whole population traverses the space, creating a high to low frequency sweep. Clustering occurs under both regimes, relying primarily on the chance union of compatible agents.

These behavioural strategies give rise to distinct population dynamics. Loitering behaviour causes large fluctuations in population size (see Figure 4, top, right). As loiterers are often the sole survivors of a population and remain close to an edge, resources accrue in the unoccupied space. If the loiterers reproduce and create offspring that are capable of accessing adjacent resources, the resource-rich

UNIFORM RESOURCE DISTRIBUTION



NON-UNIFORM RESOURCE DISTRIBUTION

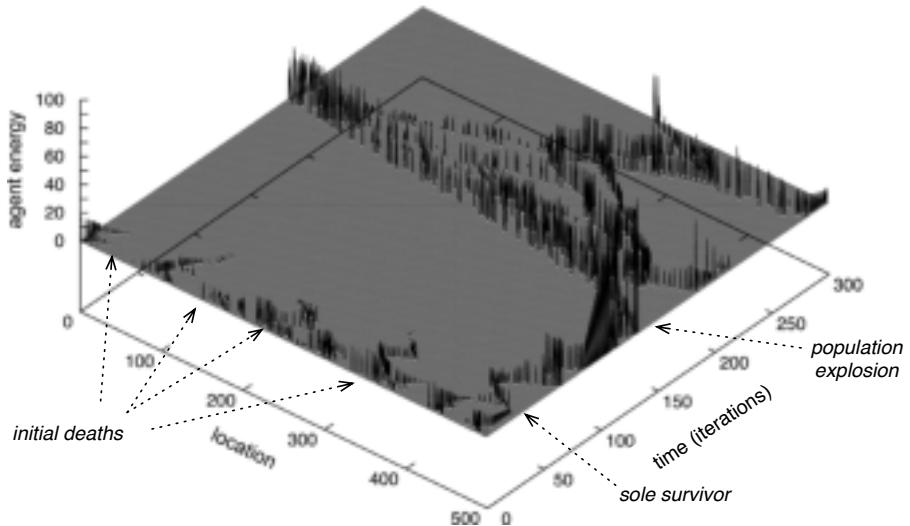


Fig. 3. Plots of energy deposits in space across 300 iterations for typical runs under uniform distribution of resources (top) and non-uniform distribution (bottom)

locations are rapidly harvested, creating a population explosion and migration. The time between these explosions and their size are positively correlated: the longer the world is empty the greater the amount of resources accrue and so the larger the population explosion that occurs. In contrast, when resources are

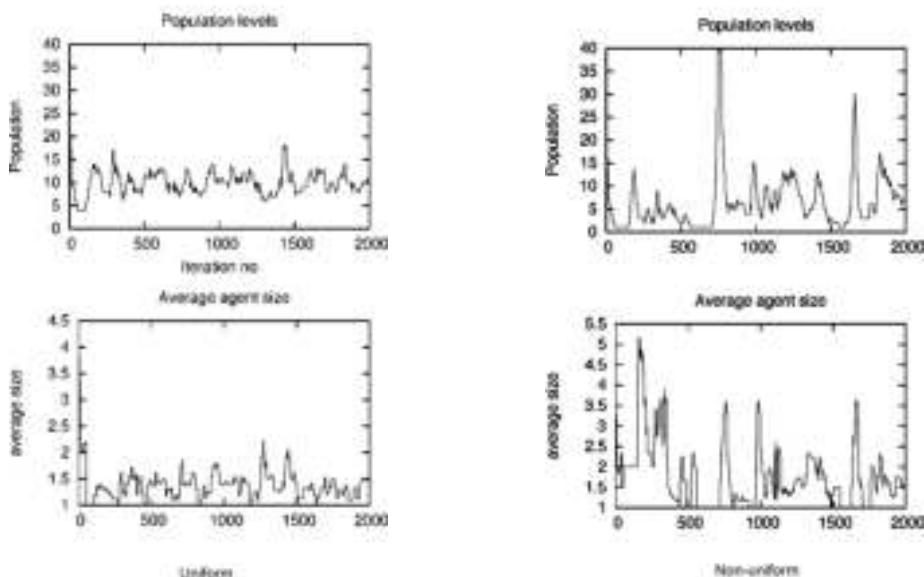


Fig. 4. Plots of population levels (top) and average agent size (bottom) for typical runs of uniformly (left) and non-uniformly (right) distributed resources

supplied uniformly to create a homogenous environment, gliding behaviour predominates, and with it a more stable population is observed (Figure 4, top, left). Sonically, these variations in population size and agent position create drones and sweeps across frequency and amplitude and are far more interesting than the static fields created when only gliding behaviour is present.

Population explosions also cause transient changes in population size distribution: statistically as the population grows, size variation increases. The effect is short-lived however as recycling is less than 100% efficient. As shown in Figure 4 (bottom) population explosions are invariably followed by transient increases in average agent size. This effect is more pronounced when resources are distributed non-uniformly (Figure 4, bottom, right). As larger agents contain more energy their presence amplifies the effect of increased population size, creating larger variations in the overall loudness of the filter.

4 Discussion

As expected, the implementation of resource recycling supports symbiotic as well as competitive survival strategies which in this context increases the range of sonic behaviour of the system. In the absence of recycling, spatially isolated agents are the most competitive. Gliding behaviour dominates, creating unstructured spectral movement. The introduction of spatial recycling means that agents can also acquire energy by occupying the frequency band at which another agent deposits its energy. In the natural world this form of symbiosis is described as

commensalism, referring to a relationship where one organism gains an advantage at no serious cost to the other. This is exemplified by carcass eaters who follow hunting animals, moving in to eat the hunters' leftovers.

Although explicable with hindsight, the emergence of loitering was unexpected at design time, but arises from the interaction of recycling with the bounded world. Sonically the variation of loitering, and clustering is much more interesting than the default gliding behaviour. In this simulation greater diversity is observed under a non-uniform, rather than uniform resource allocation procedure. Not only does this promote loitering, but the heterogenous structure created seems to amplify variation at the population level, further enhancing the sonic dynamics generated.

4.1 Future Work

In the current model, resources are undifferentiated: any agent can access any resource at any location that it occupies. Similarly, incoming and recycled resources are nutritionally equivalent. This was sufficient to demonstrate the emergence of a variety of survival strategies. The next obvious step is to model different resource types and distinguish agents according to their preference or efficiency for processing a particular type of resource. Sonically this would mean constraining agent's nutritional requirements to specific frequency bands, or samples with distinct spectro-morphological signatures. We hope that this would build a more realistic food web and increase the sonic potential of the model from the present study to a fully fledged sound work.

5 Conclusion

Filterscape illustrates how energy recycling can increase synchronic diversity in a sonic ecosystem: in this case by creating a sonic environment that supports mutualistic (clustering) as well as competitive survival strategies. Experimental manipulation of resource allocation demonstrates how the appearance of these strategies can be selectively influenced by altering basic structural properties of the world. These results support the hypothesis that heterogenous environments can support a wider range of dynamics than homogenous spaces. The ecosystemic approach is formative, but on-going experiments suggest this to be a fertile route for those interested in the emergent or collaborative potential of evolutionary art.

Acknowledgements. This research is supported by Australian Research Council Discovery Project grant DP0772667. Thanks to Jon McCormack for valuable discussions and comments on early drafts.

References

1. DiScipio, A.: Sound is the interface: From interactive to ecosystemic signal processing. *Organised Sound* 8(3), 269–277 (2003)
2. Bowers, J.: Improvising Machines: Ethnographically Informed Design for Improvised Electro-Acoustic Music. *ARiADATexts* (4) (2002)

3. Waters, S.: Performance ecosystems: Ecological approaches to musical interaction in. In: Proceedings of EMS (2007)
4. McCormack, J.: Artificial ecosystems for creative discovery. In: Thierens, D., et al. (ed.) GECCO, pp. 301–307 (2007)
5. Dorin, A.: The virtual ecosystem as generative electronic art. In: Raidl, G.R., Cagnoni, S., Branke, J. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 467–476. Springer, Heidelberg (2004)
6. Dorin, A.: Artificial life, death and epidemics in evolutionary, generative electronic art. In: Rothlauf, F., et al. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 448–457. Springer, Heidelberg (2005)
7. Eldridge, A., Dorin, A., McCormack, J.: Manipulating artificial ecosystems. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 392–401. Springer, Heidelberg (2008)
8. Laland, K.N., Odling-Smee, F.J., Feldman, M.W.: Evolutionary consequences of niche construction and their implication for ecology. *Proceedings of the National Academy of Science* 96, 10242–10247 (1999)
9. Epstein, J.M.: Growing Artificial Societies: social science from the bottom up. Brookings Institution, Washington (1996)
10. Werner, G.M., Todd, P.M.: Too many love songs: Sexual selection and the evolution of communication. In: Husbands, P., Harvey, I. (eds.) Fourth European Conference on Artificial Life, pp. 434–443. MIT Press/Bradford Books, Cambridge (1997)
11. Dahlstedt, P., Nordahl, M.G.: Living melodies: Coevolution of sonic communication. *Leonardo* 34(3), 243–248 (2001)
12. Ray, T.: An evolutionary approach to synthetic biology: Zen and the art of creating life. *Artificial Life* 1(1), 195 (1994)
13. Begon, M., Townsend, C., Harper, L.: Ecology: From Individuals to Ecosystems, 4th edn. Blackwell Publishing, Malden (2007)
14. Dorin, A., Korb, K.B.: Building virtual ecosystems from artificial chemistry. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS, vol. 4648, pp. 103–112. Springer, Heidelberg (2007)

Evolved Ricochet Compositions

Gary Greenfield

University of Richmond, Richmond VA 23173, USA

ggreenfi@richmond.edu

<http://www.mathcs.richmond.edu/~ggreenfi/>

Abstract. We consider evolutionary art based on the ricochet art-making technique. With this technique, a sequence of line segments defined by particles moving within the interior of a polygon is developed into a geometric composition by virtue of the fact that reflection (the ricochet) is used to ensure that whenever a particle meets an existing line segment it does not cross it. There is also a rule for filling some of the interior polygons that are formed by particle trajectories based on line color attributes. We establish a genetic infrastructure for this technique and then consider objective measures based on ratio statistics for aesthetically evaluating the results. For the special case of four particles in motion within a square we also examine fitness landscape questions.

1 Introduction

We consider an evolutionary art platform whose generative component is based on the ricochet art-making technique. As formulated by de Kok et al. [4], one generates a ricochet composition by positioning a particle on each edge of a square and then assigning to each particle an initial direction vector pointing toward the interior of the square. In sequence, each particle follows the straight line path determined by its direction vector until it meets an existing line segment, whence it stops and calculates a new direction vector by the principle of reflection (the ricochet). In this manner, the sequence of line segments determined by particle trajectories within the square develops into a geometric composition. Particles are assigned colors to help distinguish between them and to add more visual interest to the composition. To further enhance the artistic effect there is a polygon fill rule: If the path traced by a particle ever meets up with itself, then the resulting k -gon that is formed should be filled in with that particle's assigned color. The closest generative scheme to this technique that we are aware of is the artificial life, agent-based moving particle simulation of Annunziato and Pierucci [2].

An example of a ricochet composition obtained using pseudo randomly generated particle attributes is shown in Figure 1. The first particle, which draws in pink, starts on the left edge, and is followed in clockwise sequence by the green, blue, and orange particles. In our scheme, a particle ceases moving if the distance traveled falls below a certain threshold or it has completed nine ricochets i.e. drawn ten line segments. Since there is nothing to prevent particles from partially retracing their paths this information can be useful when trying to follow



Fig. 1. An example of a ricochet composition within a square whose initial edge locations for the particles and initial direction vectors were pseudo randomly generated

the trajectory of a particle. In Figure 1 this path retracing phenomenon occurs for both the pink and blue particles. Moreover, this knowledge also helps reveal that the blue particle in Figure 1 has hidden line segments that are obscured by polygon fill.

We explore two themes in this paper: (1) automating the search for interesting ricochet compositions by designing fitness functions for use with evolutionary computation, and (2) attempting to understand the underlying fitness landscapes induced by our fitness functions. It is of interest to note that particle trajectory problems, and similar billiard path problems, are extremely difficult to analyze mathematically as evidenced by the recent work of Baxter and Umble involving analysis for particle trajectories within triangles [3], and Levi and Tabachnikov involving similar analysis within ellipses [7].

The organization of this paper is as follows. In section 2 we carefully describe our ricochet composition implementation. In section 3 we introduce our evolutionary framework. In section 4 we discuss the use of fitness functions and provide examples. In section 5 we consider fitness landscape issues. In section 6 we present our conclusions.

2 Our Ricochet Implementation

To establish our ricochet framework, we reverse engineered the generative art process formulated by de Kok et al. [4] for their ricochet compositions. In their paper, de Kok et al. only consider ricochet compositions where the particles used to establish the sequences of line segments, or orbits, are initially assigned to distinct edges of a *square*. We generalized this so that $n \geq 3$ particles can be assigned to distinct edges of a (convex) n -gon. Moreover, by using parametric equations for the lines that determine line segments, our resulting compositions are scale free in the sense that they can be faithfully rendered onto digital canvases ranging from 200×200 pixels up to 1000×1000 pixels.

Our implementation accomplishes polygon fill by using a recursive flood-fill algorithm in order to verify whether or not all the pixels on the boundary of a polygon were determined by the same particle. We do not know if there is an alternative, elementary way to identify polygons which must be filled. This is an interesting topological question that is further confounded by demanding that polygon fill take place, not in the context of the idealized plane \mathbb{R}^2 of mathematics, but in the context of the ricochet composition's digital image representation that is defined by pixels.

Since particles are initially placed on the edges of a convex polygon, by using parametric equations for each the polygons edges, and orienting the initial unit direction vectors for the particles so that they consistently point inwards, a particle initially assigned to an edge is wholly determined by its distance from a distinguished vertex of that edge, and the angle it will depart that edge from. More precisely, if the edge from x to y is oriented so that the particle is located at $x + t(y - x)$, where $0 \leq t \leq 1$, α is the angle in degrees the vector $y - x$ makes with the horizontal axis, and a where $0 \leq a \leq 180$ is the angle in degrees the direction vector should make with the edge, then $(\cos(a+\alpha), \sin(a+\alpha))$ is the unit direction vector that defines the trajectory the particle should depart the edge from.

To implement the ricochet, we store the end points of all line segments for all orbits generated so far, and the end points of all of the n -gon's edges. Given a point and a direction we measure the straight line distance from the point to each segment, select the shortest distance, and then calculate the intersection point and unit reflection vector using that shortest distance line segment. Some care must be exercised to ensure that the shortest distance is really to one of the existing segments and not to its complement in the extended line it defines. That is, if a particle currently at p with direction heading d intersects a segment whose endpoints are x and y , then we solve $p + fd = x + g(y - x)$ for f and g using Cramer's rule, and check to make sure that $0 \leq g \leq 1$. Next, assuming without loss of generality that the segment determined by x and y is the desired one, we set $l = -d$ and choose the unit normal n to the line segment determined by x and y that satisfies $l \cdot n > 0$ i.e. such that l and n both lie in the same half plane determined by the line segment. Finally, we find the reflection vector r which will serve as the new unit direction vector using the well-known [1], or easily derived, equation $r = 2(l \cdot n)n - l$.

3 Our Evolutionary Framework

A *genome* for an n -gon ricochet composition is $(t_1, a_1; t_2, a_2; \dots; t_n, a_n)$ where t_i and a_i determine the initial edge location and direction vector of the i -th particle according to the conventions of the previous section. Treating each of the pairs (t_i, a_i) as genes makes it straightforward to implement the one-point crossover and point mutation operators we use for our artificial genetics. For convenience, to avoid degenerate cases, we restrict each t_i to lie in the interval $[0.1, 0.9]$ and each angle a_i , when measured in degrees, to lie in the interval $[10, 170]$.

4 The Use of Fitness Functions

With the preliminaries dispensed with, we are prepared to investigate the use of the genetic algorithm to evolve ricochet compositions for various regular n -gons according to different fitness criteria. Without prejudice, we use populations of size 36, replace the least fit 20 genomes after each generation by mating and mutating randomly selected pairs from the remaining breeding population of size 16, and allow the genetic algorithm to run for 40 generations. We settled on 40 since we observed that with this population size the fitness values stabilized by then. We let a_i denote the total area filled by the i -th particle, and e_i the path length of the i -th particle. The subtlety here is that we may have to adjust the *total* path length traveled by the particle to account for the fact that certain segments of the path may be hidden due to polygon fill in order to obtain a path length value that more accurately represents what we actually see when we visually inspect the composition. In the sequel our objective will always be to maximize ricochet composition fitness using fitness functions in these arguments.

To understand why the choice of fitness function is so significant, first consider the $n = 4$ composition in Figure 2 — shown both unfilled and filled — that was evolved using fitness proportional to $\sum_i \frac{a_i}{1+e_i}$. Because of the interdependence between a_i and e_i compositions evolved using this fitness criterion will often be described as having a feeling of balance. Perhaps this is because the local maxima found using a genetic algorithm will usually be achieved by convergence to a uniform ratio for each term in the sum.

Next, consider the composition in Figure 3 evolved using fitness $\sum_i e_i$. While it is easy to understand why the composition looks the way it does, the relevant observation we wish to make here is that for lack of a better word, the composition’s “style”, a style that is more harsh and angular than the previous example,

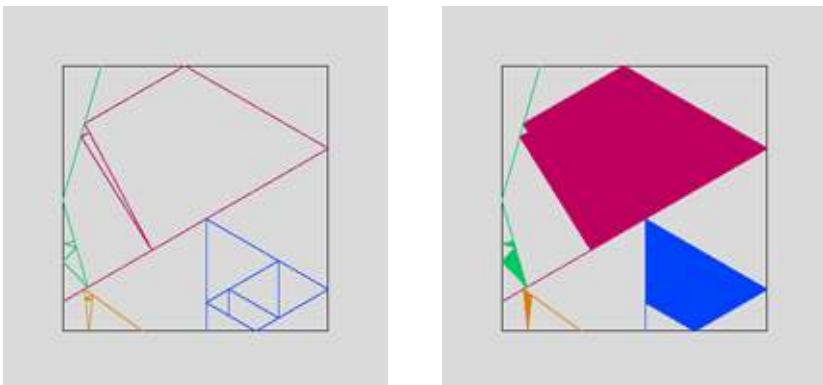


Fig. 2. An evolved ricochet composition, shown both before and after polygon fill, whose fitness was calculated using a sum of ratio terms of the form $a_i/(1 + e_i)$ where a_i and e_i are the path length and filled area of the i -th particle

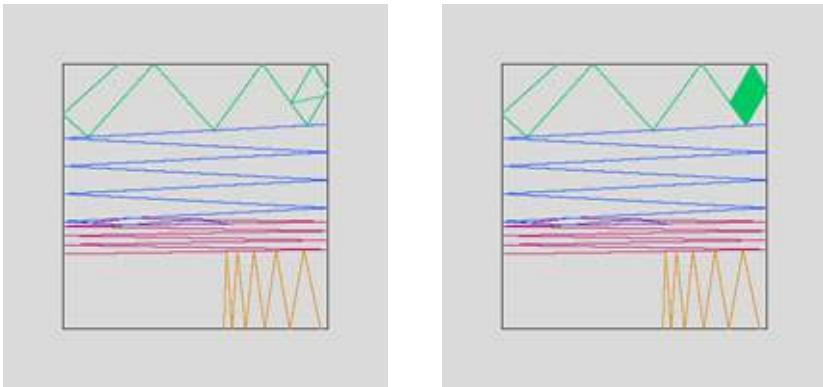


Fig. 3. An evolved ricochet composition, shown both before and after polygon fill, whose fitness was calculated base on total particle path length

has been adversely affected by the fitness evaluation method chosen. We remark that Figure 3 immediately suggests how one might characterize what maximally fit compositions in this style should look like.

One of the more contentious issues involving questions about aesthetics in generative art is whether imposing some degree of symmetry on the images that will be generated is desirable. Be that as it may, we experimented with a rotational symmetry measure for the case of a square ($n = 4$) by first decomposing the square into four isosceles triangles, or wedges, whose apexes are at the center and each of whose bases are one of the edges. After numbering them sequentially from 0 to 3, we define for $0 \leq i < j \leq 3$, $s(i, j)$ to be the number of mismatches between filled and unfilled aligned pixels after wedge i is rotated to coincide with wedge j . Thus, if $s(0, 2) + s(1, 3)$ were zero, the image would have perfect “half turn symmetry”. Sadly, we report that all fitness functions we designed incorporating such terms were judged aesthetic failures. For example, Figure 4 shows a run where the effort to achieve half turn symmetry led to one particle dominating the composition. Another contentious issue is selecting the “best” evolved examples from among several runs using the same fitness function. When comparing the results from short lists of candidate fitness functions, we attempted to mitigate this factor by subjectively ranking the candidates on the list on the basis of only two runs per candidate.

The fitness measure we judged to be most satisfying was one maximizing the ratio statistic $\min_i \frac{a_i}{1+e_i}$. Evolved examples in this style for $n = 5$ and $n = 6$ are shown in Figure 5. Note that gray and purple were added as the two new particle colors respectively. Although in general we found that we preferred compositions evolved using this criterion, we found that as n increased this fitness measure began to fail most likely because there was insufficient room for the particles to establish trajectories yielding fine details that would make the composition more visually appealing. This realization occurred by comparing sets of $n = 16$ ricochet compositions made using alternating black and blue particles that were

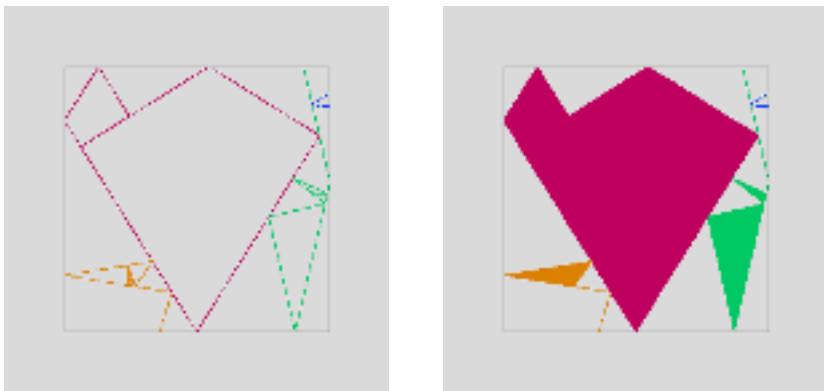


Fig. 4. An evolved ricochet composition, shown both before and after polygon fill, whose fitness was determined by how close it came to achieving half turn symmetry

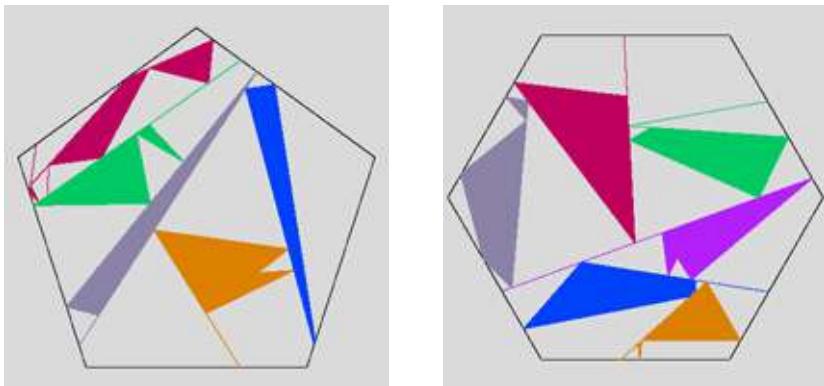


Fig. 5. Evolved $n = 5$ and $n = 6$ ricochet compositions. Both were evolved by trying to maximize the minimum particle ratio quantity $a_i/(1 + e_i)$.

pseudo randomly generated with sets that were evolved. Figure 6 pairs a random black and blue composition with an evolved black and blue composition.

5 The Fitness Landscape

The only generative art example we are aware of where the fitness landscape was identified through the use of exhaustive search was done by Krawczyk [5] [6]. It involved a special type of strange attractor. More recently Tsang [8] used large random samples of fitness values to identify characteristics of the fitness landscape in order to help tune the parameters of the genetic algorithm used to evolve aesthetic visualizations of particle trajectories arising from simulating the N-body problem. Tsang interpolated between high fitness genomes and used

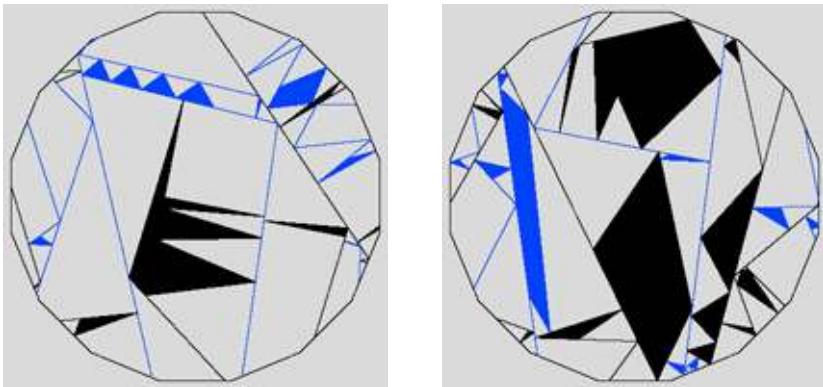


Fig. 6. Left: An $n = 16$ “black and blue” ricochet composition generated by pseudo randomly generating the initial conditions. Right: An evolved $n = 16$ “black and blue” ricochet composition obtained by trying to maximize the minimum particle ratio quantity $a_i/(1 + e_i)$.

the resulting sequence of fitness calculations to measure the “convexity” between the fitness peaks. Additionally, Tsang relied on skewness and kurtosis to identify “long tails” in fitness distributions.

The $n = 4$ ricochet composition case where four particles are in motion within the square offers us the opportunity to explore fitness landscapes from a different point of view. In this situation large random samples are more likely to be representative of the underlying fitness distributions. Moreover quasi-systematic sampling can be used to reveal how successfully we can search in these spaces. That is, it should reliably be able to reveal how successfully we will be able to “explore and exploit” using the genetic algorithm.

We considered five fitness functions. These functions are listed in Table 1. We first ran a benchmark series of experiments by tabulating the fitness results for four sets of 4096 randomly generated genomes for each of these fitness functions. Next, we quasi-systematically examined the fitness landscape as follows: we fixed the gene for the first particle by setting it to one of sixteen ordered pairs by taking edge location parameter t to be one of .15, .35, .55, .75 and direction angle a to be one of 15, 55, 95, 135 and then generated fitness data for all genomes consisting of the 16 possible settings for the genes associated with the other three particles. In this way we obtained 16 data sets each with 4096 observations.

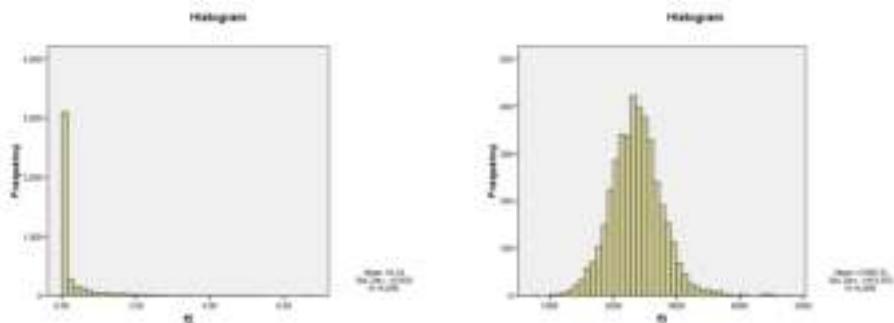
Since for all these functions we maximize fitness, Table 2 shows the maximum values we obtained for each of the fitness functions from our four benchmark runs. We note that the fitness for the image in Figure 2 using fitness function f_1 is 10.07 and the fitness for the image shown in Figure 3 using fitness function f_3 is 5914. Both values are reasonable. The values for the mean, standard deviation, median, skewness, and kurtosis were stable across our benchmark runs. Histograms (see Figure 7) reveal that the distribution for f_3 is approximately normal, which is not surprising in view of the central limit theorem, but also that the distribution for f_2 , our preferred fitness function, most closely resembles

Table 1. Table of fitness functions whose fitness landscapes were considered

Function	Formula
f_1	$1/4 \sum_i \frac{a_i}{1+e_i}$
f_2	$\min_i \left\{ \frac{a_i}{1+e_i} \right\}$
f_3	$\sum_i e_i$
f_4	$\sum_i a_i$
f_5	$\min_i \{e_i\}$

Table 2. Table of maximum fitness values from four benchmark runs of 4096 randomly generated genomes

Function	M_1	M_2	M_3	M_4
f_1	10.96	11.68	13.28	10.73
f_2	7.24	9.05	8.20	6.76
f_3	5710	4912	4952	4779
f_4	38426	41433	42127	39692
f_5	3881	4702	5163	4027

**Fig. 7.** Histograms of fitness values obtained from one of our four benchmark random samples of 4096 $n = 4$ genomes under fitness functions f_2 (left) and f_3 (right)

an exponential distribution with kurtosis in the 23 to 28 range. In fact the percentage of genomes with nonzero fitness under f_2 for the benchmark runs was $49.5\% \pm 1\%$. A similar statement holds for f_5 . This is easily explained by the fact that they will yield fitness values that are nonzero only if all four particles have self intersecting trajectories, whence they do bound areas of interior polygons. From this unsophisticated analysis we thereby establish that the fitness landscapes for f_2 and f_5 are both rugged and sparse.

We now turn our attention to the systematic sampling results. The statistical results we obtained for f_1 and f_3 were consistent with those of the random data sets. This is not surprising in view of the distributions of fitness values we observed. Therefore we concentrated on f_2 . The histograms of the fitness values

using f_2 for all 16 systematically generated data sets were “binned” meaning that fitness values were concentrated within small intervals. Moreover, the kurtosis varied dramatically from a low of 10.426 to a high of 33.84 with six values below 20 and three above 30. Similarly, the maximums ranged from 5.21 to 8.85 and the percentage of nonzero fitness values varied from a low of 25.2% to a high of 38.47%. We interpret this data as telling us that for the fitness landscape induced by f_2 , local maxima are isolated and they are of different qualities. The four highest fitness values obtained using these data sets for each of f_1 , f_2 , and f_3 are shown in Table 3.

Table 3. Table showing the four maximum fitness values within sixteen systematically generated fitness data sets

Function	M_1	M_2	M_3	M_4
f_1	12.30	12.24	12.18	12.08
f_2	8.85	8.35	8.23	7.84
f_3	4231	4201	4231	4088

6 Conclusions

Ricochet compositions provide a platform for investigating a simpler generative art system than is usually considered in the literature. Given the goal of evolving visual images with aesthetic properties, we chose to work with this platform because the immediate consequences of fitness function design could be more easily demonstrated and because questions about the fitness landscape induced by fitness functions appear to be more tractable.

We feel one of the fundamental obstacles researchers in the area of visual evolutionary art face is validating their methodology i.e. providing evidence for assertions about the mechanisms used and the success achieved in identifying aesthetic images. In particular, there is a need to bind fitness function design with tangible aesthetic attributes. To address such issues, new techniques for examining and assessing the conditions under which the underlying genetic algorithms within evolutionary art systems behave, and what their limitations are, are needed. In this paper, we have attempted to show how elementary statistical sampling and analysis may be of potential use in this area.

References

- Angel, E.: *Interactive Computer Graphics A Top-Down Approach with OpenGL*, 4th edn. Pearson Addison-Wesley, Upper Saddle River (2006)
- Annunziato, M., Pierucci, P.: Relazioni emergenti: experiments with the art of emergence. *Leonardo* 35(2), 147–152 (2002)
- Baxter, A., Umble, R.: Periodic orbits for billiards on an equilateral triangle. *Amer. Math. Monthly* 115(6), 479–491 (2008)
- de Kok, I., Lucassen, T., Ruttkay, Z.: Ricochet compositions. In: Sarhangi, R., Barrallo, J. (eds.) *BRIDGES 2007 Conference Proceedings*, pp. 177–180 (2007)

5. Krawczyk, R.: Dimension of time in strange attractors. In: Barrallo, J., et al. (eds.) BRIDGES/ISAMA 2003 Conference Proceedings, pp. 119–126 (2003), <http://www.iit.edu/~krawczyk/rjkbrdg03.pdf>
6. Krawczyk, R.: The ghostly imagery of strange attractors. In: Sarhangi, R., Sequin, C. (eds.) BRIDGES 2004 Conference Proceedings, pp. 333–336 (2004), <http://www.iit.edu/~krawczyk/rjkbrdg04a.pdf>
7. Levi, M., Tabachnikov, S.: The Poncelet grid and billiards in ellipses. Amer. Math. Monthly 114(10), 895–908 (2007)
8. Tsang, J.: Evolving trajectories of the n-body problem. In: 2008 IEEE World Congress on Computational Intelligence (CEC 2008), pp. 3726–3734 (2008) (Digital Object Identifier 10.1109/CEC.2008.4631302)

Life's What You Make: Niche Construction and Evolutionary Art

Jon McCormack and Oliver Bown

Centre for Electronic Media Art

Monash University, Clayton, Victoria 3800, Australia

Jon.McCormack@infotech.monash.edu.au, Oliver.Bown@infotech.monash.edu.au

Abstract. This paper advances new methods for ecosystemic approaches to evolutionary music and art. We explore the biological concept of the niche and its role in evolutionary dynamics, applying it to creative computational systems. Using the process of niche construction organisms are able to change and adapt their environment, and potentially that of other species. Constructed niches may become heritable environments for offspring, paralleling the way genes are passed from parent to child. In a creative ecosystem, niche construction can be used by agents to increase the diversity and heterogeneity of their output. We illustrate the usefulness of this technique by applying niche construction to line drawing and music composition.

1 Introduction

A ‘grand challenge’ for evolutionary music and art (EMA) is “To devise unique kinds of evolutionary ‘software instruments’ that offer the possibility of deep creative engagement and enable the creative exploration of generative computational phase-spaces.” [1]

A major paradigm of the past for EMA has been the interactive genetic algorithm (IGA) whereby the difficulty of defining a machine-representable fitness function for aesthetics is bypassed in favour of human-based selection. In this scenario, the user becomes a kind of ‘pigeon breeder’, searching, often blindly, for some hopeful monster within the limitations imposed by aesthetic selection [2].

In recent years a new paradigm has emerged, that of the *creative ecosystem* [1,3,4,5]. In this approach, the focus is shifted from aesthetic fitness evaluation – either automated or manual – to the design of specific components and their interactions to form a dynamic ecosystem. Creative ecosystems may optionally incorporate evolution. The important features of this approach can be summarised by three observations:

- An ecosystem consists of *components* with carefully designed *interactions* between themselves and their environment;
- The ecosystem operates and is conceptualised within the generative medium itself, for example a sonic ecosystem operates in the medium of sound, rather than being a sonification of some other process;

- Components within an ecosystem are interconnected in such a way that they can modify their (biotic or a-biotic) environment, often to their own benefit or that of their descendants.

Creative ecosystems exhibit the characteristic features of real ecosystems: heterogeneity, diversity, mutualism, stability under change, and complex feedback loops [6]. The ecosystem approach mirrors the realisation in Biology that evolution is more than just selection and adaptation – organisms not only adapt to environments, they actively change them in order to create local environments better suited to them and their offspring.

While ‘ecosystemics’ represents a promising new approach, a challenge remains to understand the classes of mechanisms most appropriate to creative applications. In this paper we look at mechanisms from Ecology, and show how they can be adapted for creative ecosystems to improve the aesthetic possibilities of generative systems. We focus on the concepts of *niches*, *niche widths*, *niche construction* and *ecosystem engineering*. We will demonstrate how these concepts can be applied to creative generative systems in order to improve the aesthetic properties and diversity of output. This will be illustrated using example line drawing and music generation systems.

1.1 Niches

Biological environments have two broad properties that determine the distribution and abundance of organisms: *conditions* and *resources*. Conditions are physiochemical features of the environment (e.g. temperature, pH, wind speed). An organism’s presence may change the conditions of its local environment (e.g. one species may modify local light levels so that other species can be more successful). Conditions can change in cycles: diurnal, annual or according to the frequency of extreme events. Conditions can also serve as stimuli for other organisms. Resources, on the other hand, are consumed by organisms in the course of their growth and reproduction. One organism may become or produce a resource for another through grazing, predation, parasitism or symbiosis, for example.

For any particular condition or resource, an organism may have a preferred value or set of values that favour its survival, growth and reproduction. One such characteristic curve is shown in Fig. 1 (left).

The complete set of conditions and resources affecting an organism represent its *niche*, which can be conceptualised as a hypervolume in n -dimensional space. As an example, for two conditions c_1 and c_2 , two different types of species relationships are shown in Fig. 1. The shaded area represents the ‘viability zone’ for the species. A species will only survive if conditions are maintained within this shaded area. A relatively large distance in any single dimension denotes a generalist *in that dimension* (s_1 is relatively generalist in c_2), specialists have small distances (s_3 is more specialised in both c_1 and c_2). This size is referred to as *niche width*, and may vary for each dimension.

Competition and other species interactions are important in determining habitat distribution. Niche differentiation can permit coexistence of species within a biotope. Higher number of species can coexist by utilising resources in different

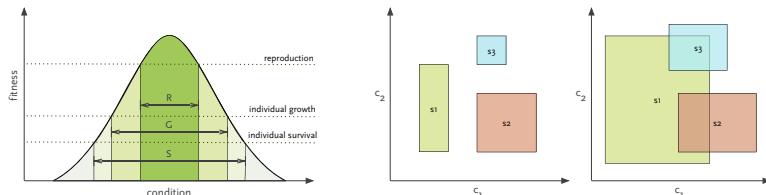


Fig. 1. Example organism viability zones for reproduction, growth and survival (left) and exclusive and overlapping niche areas for a two-dimensional set of conditions

ways. It is reasonably well understood in Biology how these mechanisms give rise to species diversity and specialisation. The challenge in EMA research is to devise useful ways of employing these mechanisms in creative contexts. Here the question is how to devise mappings between conditions and resources, and establish trade-offs for an individual's survival based on tolerances to specific conditions in order to enhance the quality and diversity of output in a creative generative system.

2 Niche Construction

Niche construction is the process whereby organisms modify their own and each other's niches. They do this by modifying or influencing their environment. Proponents of niche construction argue for its importance in understanding how the evolutionary process works in nature [7]. By modifying their niche, organisms may provide a heritable environment for their offspring. Hence niche construction can create forms of feedback that modify the dynamics of the evolutionary process, because ecological and genetic inheritance co-influence in the evolutionary process. Computational models of niche construction show that it can influence the inertia and momentum of evolution and introduce or eliminate polymorphisms in different environments [8]. Other models have demonstrated that a simple niche constructing ecosystem can support homeostasis and bi-stability similar to that of Lovelock's popular *Daisyworld* model [9].

Whereas standard evolutionary algorithms tend to converge to a single (sub)-optimum, niche construction can promote diversity and heterogeneity in an otherwise fixed and homogeneous evolutionary system. In systems where the design of an explicit fitness function may be difficult or impossible (as in many EMA systems), niche construction provides an alternate mechanism to explore a generative system's diversity over an IGA. An ecosystemic approach to creative systems does not necessarily parallel, or serve as a replacement to, traditional Evolutionary Computing methods, as creative ecosystems are not defined by a single algorithm or method. An important consideration is that the ecosystem must be developed specifically to the domain and creative system desired. A process such as niche construction serves as a 'design pattern' [10] to help facilitate the design of such systems. In order to illustrate the utility of niche construction we will now describe two different experiments where niche construction influences the structure and variation of the creative artefacts produced.

2.1 Line Drawing

In order to demonstrate the power of niche construction in a creative context, we begin with a simple, agent-based line drawing program and show that by adding a niche constructing component, the aesthetic sophistication and diversity of drawings created by the system increases significantly.

Our system is inspired by Mauro Annunziato's *The Nagual Experiment* [11], principally because this system actually produced interesting drawings from an artistic perspective, but also because of its simplicity. The system consists of a number of haploid drawing agents that move around over a two-dimensional drawing surface – initially a blank, white canvas. Agents move over the surface, leaving an ink trail as they go. If an agent intersects with any existing trail, either drawn by itself or by another agent, it dies. Agents may produce offspring that grow out from their parent at their time of birth.

The characteristics of the path an agent chooses to draw is determined by its genes. The actual path is determined by a stochastic process. An agent's genome has the following alleles, each represented as a normalised floating point number:

curvature controls the rate of curvature of the line. Varies from a straight line (0) to a maximum curvature rate (1);

irrationality controls the rate and degree of change in the rate of curvature according to a stochastic algorithm (see Fig. 2);

fecundity (f), the probability of the agent reproducing at any time step. New agents are spawned as branches from the parent;

mortality the probability of the agent dying at any time step;

offset the offset angle of child filaments from the parent;

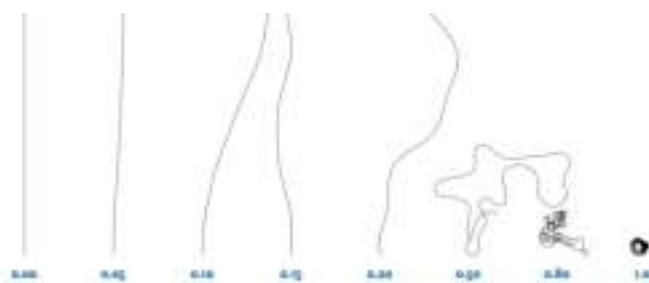


Fig. 2. Individual line drawing agents with different measures of irrationality

The canvas is seeded with a number of drawing agents who proceed to draw according to the characteristics described by their genes. A drawing is complete when all the agents have died. The system produces a variety of interesting drawings, due in part to the interaction between agents and the lines they draw (Fig. 3). Initial 'founder' lines can carve out spaces and prevent other lines from drawing into them due to the rule that line intersection causes death.

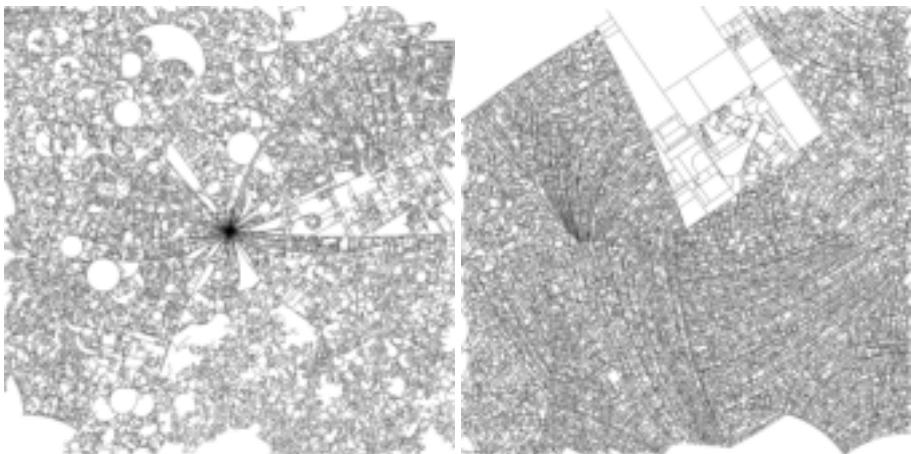


Fig. 3. Two example drawings produced by the agent system (no niche construction)

While the drawings are interesting, they are largely homogeneous, both in terms of the style and overall tonal density observed in the images produced. By adding a niche construction process the images become much more heterogeneous and exhibit greater aesthetic variation.

The Drawing Niche Construction Model. To add niche construction to the drawing model, each agent is given an additional allele in its genome: a local density preference δ_i (a normalised floating point number). This defines the agent's preference for the density of lines already drawn on the canvas in the immediate area of its current position, i.e. its niche (Fig. 4). In a preferred niche, an agent is more likely to give birth to offspring and has a better chance of survival. As children inherit their parent's genes they are more likely to survive as they have a similar density preference. So in a sense, parents may construct a niche and pass on a heritable environment well-suited to their offspring.

For each agent, i , δ_i defines it's preferred niche. Local density (the ratio of inked to blank canvas per unit area) is measured over a small area surrounding the agent at each time step. Proximity to the preferred niche determines the probability of reproduction of new agents, given by: $Pr(\text{reproduction}) = f_i \cdot \cos(\text{clip}(2\pi(\Delta_{p_i} - \delta_i)), -\frac{\pi}{2}, \frac{\pi}{2})$, where Δ_{p_i} is the local density around the point p_i , the agent's location. f_i is the agent's fecundity and 'clip' is a function that limits the first argument to the range specified by the next two. Being in a non-preferred niche also increases the chances of death.

Agents begin with a low density preference, uniformly distributed over $[0, 0.25]$. Beginning the drawing on a blank canvas means that only those agents that prefer low density will survive. As the drawing progresses however, more ink is added to the canvas and agents who prefer higher densities will prosper. At each birth the agents genome is subject to the possibility of random mutation (proportional to

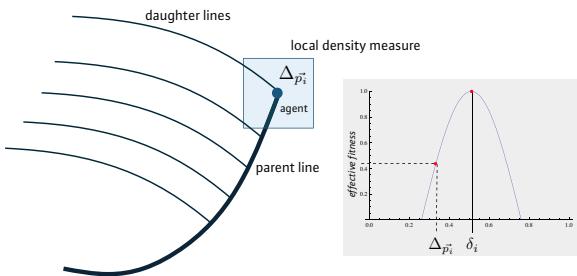


Fig. 4. The niche construction mechanism for drawing agents, who try to construct a niche of local density that satisfies their genetic preference

the inverse of the genome length), allowing offspring to adapt their density preference and drawing style as the drawing progresses. Eventually the population becomes extinct, since high density favouring agents don't have much room to move, and the drawing finishes. Some example drawings are shown in Fig. 5. Notice the greater stylistic variation and heterogeneity over the images shown in Fig. 3.



Fig. 5. Two example drawings produced with the addition of niche construction

2.2 Musical Niche Construction

The *RiverWave* model is a sonic ecosystem in which agents contribute to the construction of an evolving additive synthesis soundscape. The agents inhabit a sonic environment which they contribute to and this environment in turn defines selection pressures for the agents. The model explores the long term evolutionary dynamics of a system in which environmental conditions and genetically determined behaviours coevolve, and demonstrates the efficacy with which an

artist-programmer can design a set of interactions that produce interesting continuous change. The model was designed to be run as a durational audio visual installation in a corridor space as part of the 2008 NetAudio Festival¹. The intention was to establish coherent behaviour that varied over short and long term time scales, that passers by would revisit to explore its varying behaviour over time.

The Environment. The environment is a toroidal one-dimensional space consisting of 300 cells, each of which produces a single sine tone along a microtonal pitch scale. Each cell stores an amplitude value for the frequency assigned to it. Together the cells produce a continuous sound which is the sum of the individual weighted sine tones. The amplitude of each cell is represented visually as the height of the line. Heights are constrained to the interval $[-1, 1]$ with negative values mapped to positive amplitudes. This amplitude value is affected by agents, and also obeys physical laws which make the line behave like an elastic string: the amplitude values for each cell decay exponentially to zero, and adjacent cells pull on each other's amplitude values in proportion to the difference between their respective values. This acts as a smoothing function on the line, which relaxes to zero in the absence of agent forces.

The environment has two natural properties that can be used as conditions to which agents are exposed: a height, y , and a gradient dy . These define the environmental conditions to which agents may be more or less adapted.

The Agent. Agents inhabit this one-dimensional environment, occupying a single cell for the duration of their lifetime. Agents apply forces to the line at their current location and at the location of the cell to their left, affecting the amplitude of these cells and their neighbours via the physical properties of the line (left of Fig. 6). Agents also have genetically determined preferences for y and dy , as well as a degree of specialism with respect to each of these conditions. Agents receive discrete fitness rewards if each of the local environmental conditions are within their preference range, with more specialist agents obtaining greater fitness rewards.

At each time step, the rewards accumulated from this environmental interaction contribute to the agents' health. Since these rewards are added independently, an agent can be poorly adapted to one niche dimension, but well adapted the other and still survive. The agent's health value decays exponentially in proportion to the number and proximity of other agents in the space (many agents nearby reduce health by a large amount, few agents far away reduce health by a small amount), and by the agent's age (older agents' health decays more rapidly). Since the agent's health can accumulate and be stored over multiple time steps, agents can survive without health gains for some time. Agents are given an initial maturation period during which they can modify their environment but cannot reproduce or die, after which an agent's health is used to determine whether it

¹ An annual festival of internet-based digital music, held at *The Shunt Vaults*, London, in October 2008. See <http://www.netaudiolondon.cc>

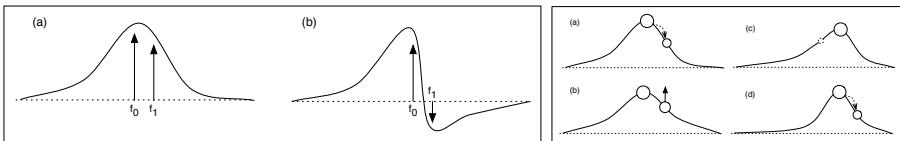


Fig. 6. Left: Agents exert two independent forces on two adjacent points along the line, which can be in the same (a) or opposite (b) directions. A smoothing function along the whole line draws adjacent points closer together over time. Right: Example of drift. Agents are adapted to slopes. Energy is represented by size. (a) An agent produces a child to its right. (b) The child gains energy because it is on a slope and pushes upwards. (c) The parent dies. It has lost energy because it is on a peak. (d) The child is now on a peak but still with high energy, and produces a new child on the slope to its left.

dies or gets to reproduce. The maturation period is useful in providing a window of opportunity for agents in poor environments to improve their conditions, thus increasing the efficacy of niche construction.

Agents also have a genetically determined offspring displacement value: the position at which their children are placed relative to their own position. This displacement property was inspired by the idea of life along a river, or in the context of seeds being carried in a prevailing wind direction, defining established geographic pathways. This genetically inherited displacement value is mutated at a low rate, but without upper limit, hence a growing population will tend to fan out along the line in discrete steps.

Examining Model Behaviour. The system demonstrated divergent evolutionary pathways, believed to be due to its niche constructing nature. Although the environment starts out flat ($y = 0$ and $dy = 0$), the randomly initialised population may be poorly adapted to that environment (preferring non-zero values) and may also inadvertently affect the environment by exerting forces on it, regardless of what kind of environment they prefer. In these initial stages, environmental modification may be unintentional, but still has the effect of determining future evolutionary pressures in an undirected manner.

Fig. 7 shows the evolution of the sound spectrum across six runs of the model with the same settings but random initial populations. A number of different population behaviours can be observed in the model, as well as sharp transitions in qualitative behaviour. Run 5 shows diagonal movement of spectral peaks, which is caused by populations drifting horizontally as children replace parents and in turn produce further descendants with the same, or similar, fixed offspring displacement. This collective behaviour may in theory be one that is reinforced through evolution, as illustrated on the right of Fig. 6: a mature agent pushes the line up to a peak and then produces a child at a slight displacement, the child therefore being born on a slope. If the child is adapted to living on slopes it will gain energy. Meanwhile, the parent, who, like the child, is also adapted to slopes, loses energy and dies. Once the parent's force on the line has gone,

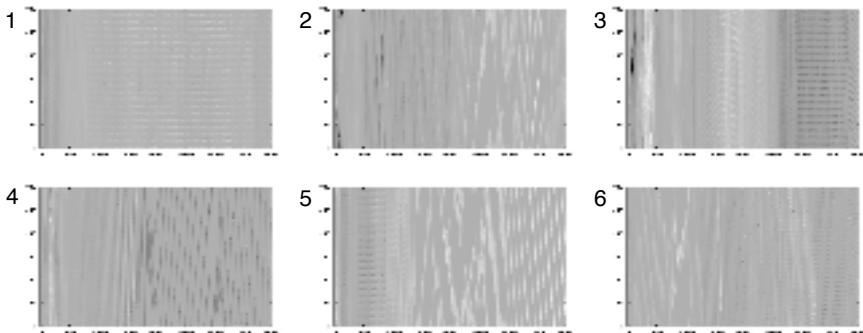


Fig. 7. Multiple runs of the same simulation, with time step on the horizontal axis, cell index on the vertical axis and amplitude represented by shading

the child's location shifts from a slope to a peak. In the meantime, the child has produced its own offspring, who now lives on a slope, repeating the cycle. At least in theory, this is a self-sustaining situation, which can be interpreted as the construction of suitable niches by parents for their offspring. However, the situation may be unstable due to alternative selection pressures, or simply because the cycle may become broken.

3 Conclusion

In this paper we have considered how niche construction as a biological phenomenon can be applied to creative ecosystems in order to increase the heterogeneity and the diversity of output in both visual and musical systems. In the visual case, we compared the same model with and without niche construction and demonstrated how niche construction drives a more heterogeneous and aesthetically diverse environment. In the audio case, we demonstrated how a niche constructing system could be designed using properties derived from the aesthetic medium – the sound spectrum – to drive ongoing evolutionary change. In this case, we did not attempt to compare models with and without niche construction; the model was directly inspired by the process of niche construction and a non-niche construction version would have made little sense. These studies demonstrate how the niche construction ‘design pattern’ facilitates a move from a predefined regime of interactions between an agent and its environment to an evolving one. Further study into the use of such mechanisms may lead to more general guidelines for how to establish rich behavioural complexity in generative artworks.

Acknowledgements

This research is supported by Australian Research Council Discovery Project grant DP0877320.

References

1. McCormack, J.: Facing the Future: Evolutionary Possibilities for Human-Machine Creativity. In: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music, pp. 417–451. Springer, Heidelberg (2008)
2. Dorin, A.: Aesthetic fitness and artificial evolution for the selection of imagery from the mythical infinite library. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, pp. 659–668. Springer, Heidelberg (2001)
3. Di Scipio, A.: ‘sound is the interface’: from interactive to ecosystemic signal processing. *Organised Sound* 8(3), 269–277 (2003)
4. Driessens, E., Verstappen, M.: Natural Processes and Artificial Procedures. Natural Computing Series. In: Design by Evolution: Advances in Evolutionary Design, pp. 101–120. Springer, Heidelberg (2008)
5. Dorin, A.: A Survey of Virtual Ecosystems in Generative Electronic Art. In: The Art of Artificial Evolution, pp. 289–309. Springer, Heidelberg (2006)
6. May, R.M.: Stability and Complexity in Model Ecosystems, 2nd edn. Princeton University Press, Princeton (2001)
7. Odling-Smee, J., Laland, K.N., Feldman, M.W.: Niche Construction: The Neglected Process in Evolution. Monographs in Population Biology. Princeton University Press, Princeton (2003)
8. Day, R.L., Laland, K.N., Odling-Smee, J.: Rethinking adaptation: the niche-construction perspective. *Perspectives in Biology and Medicine* 46(1), 80–95 (2003)
9. Dyke, J.G., McDonald-Gibson, J., Di Paolo, E., Harvey, I.R.: Increasing complexity can increase stability in a self-regulating ecosystem. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 133–142. Springer, Heidelberg (2007)
10. Gamma, E.: Design patterns: elements of reusable object-oriented software. Addison-Wesley professional computing series. Addison-Wesley, Reading (1995)
11. Annunziato, M.: The nagual experiment,
<http://www.plancton.com/papers/nagual.pdf>

Global Expectation-Violation as Fitness Function in Evolutionary Composition

Tim Murray Browne^{1,*} and Charles Fox²

¹ Computing Laboratory
University of Oxford

tim.murraybrowne@elec.qmul.ac.uk

² Adaptive Behaviour Research Group
University of Sheffield
charles.fox@gmail.com

Abstract. Previous approaches to Common Practice Period style automated composition – such as Markov models and Context-Free Grammars (CFGs) – do not well characterise global, context-sensitive structure of musical tension and release. Using local musical expectation violation as a measure of tension, we show how global tension structure may be extracted from a source composition and used in a fitness function. We demonstrate the use of such a fitness function in an evolutionary algorithm for a highly constrained task of composition from pre-determined musical fragments. Evaluation shows an automated composition to be effectively indistinguishable from a similarly constrained composition by an experienced composer.

1 Introduction

Mechanical composition in the Common Practice Period (CPP) style dates at least to Mozart’s dice game, which presented 170 bar-length musical fragments and a first-order Markov transition matrix for assembling them into sequences [20]. The Markovian approach to automated composition has continued to the present day, and n -gram Markov models may be found in both generative [6] and evaluative [12] stages of composition. An alternative class of automated composition methods is the use of context-free grammars (CFGs) [4,9].

Both Markovian and CFG approaches can produce ‘correct’ sounding compositions, but on extended listening can still sound dull and formulaic. Both approaches tend to lay out notes without considering the global context which is, of course, everything that is heard before *and after* a given note. The need for such a global structure in algorithmic composition is often identified as an open research question [8,13]. A recent state-of-the-art evolutionary model for rhythm generation [7] identified but circumvented this issue by composing only rhythms and taking their cues from global structures provided in human-composed tracks of the composition.

* Tim Murray Browne is now a research student at Queen Mary University of London.

A new idea in automated composition is to provide a global structure by systematically violating listeners' expectations. Minsky [15] presented the idea of listening to music as exploring a scene, where we absorb different ideas and infer their structural relations. To maintain interest we must always be learning new ideas and relating them to previous ones, which gives some clue as to why Markov and rule-based processes that imitate what has already been heard produce dull results. The problem is that, whilst Markov chains and grammatical parsing are effective models of the *local* listening process, they are not so of the generative process and global perception. In this view, composing is about *manipulating* the listening processes and so whilst the composer must be able to evaluate their composition by listening, there are distinct mechanisms for creating music [18]. The listener needs to be taught ideas then surprised when they are broken or extended. Abdallah and Plumbley [1] recently modelled listening with information measures of a model-based observer. The Kullback-Leibler (KL) divergence, between before and after a note is heard, measured the average rate at which information arrives about the future. Applied to minimalist music, this showed that the listener is constantly receiving *new* information about the rules the music is following, and so learning its structure. This raises questions about the completeness of reductionist approaches to composing music such as CFGs. A composition cannot be broken down into ever smaller sections as ideas that are presented early on must develop and interact, and any part of a composition is always heard in the context of its absolute position within the piece. Analogously to linguistic generation, we may use knowledge of grammar to produce grammatically correct sentences, and with some advanced linguistics perhaps even make them coherent, but by themselves they does not produce interesting or moving prose, let alone poetry.

We present a new method for aesthetic analysis of harmony based on emotional tension, and its use as a fitness function in an evolutionary algorithm for a simplified compositional task. We combine Abdallah and Plumbley's notion of expectation-violation with a classic neural-network listening model [2]. Both of these models were intended for analysis and are here novelly redeployed for use in creative composition. Our compositional task is deliberately constrained, and the present work is intended to illustrate the use of the fitness function and suggest its incorporation into more advanced existing composition systems – we do not intend the simplified compositional task to be a full compositional system in itself.

2 The Global Structure of Expectation-Violation

To form an aesthetic judgement we require tools that work at a human level, emotionally and semantically. Meyer describes the aesthetic experience of listening to music as the *manipulation* of expectation involving the buildup of tension and subsequent release [14]. We will follow this approach and adopt a formal measure of tension based on Abdallah and Plumbley's unmet expectation of resolution. The importance of addressing tension and resolution in composition

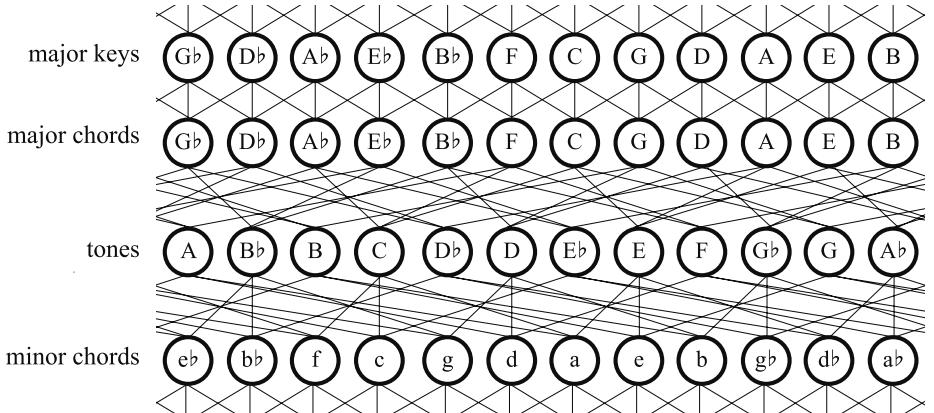


Fig. 1. Bharucha’s network, which we redeploy for tension measurement. Links at the sides wrap around.

systems has been noted in recent research as well as the analysis of such qualities with respect to their context [17].

Throughout CPP music many common chord progressions occur. If we play the beginning of a common chord sequence the listener will experience anticipation of which chord is coming next. While we maintain this expectation we build up tension. To prevent our listener from becoming bored we require consistent build up and release of tension and also, to prevent our piece from becoming segmented, we require an increase in the intensity of these build-ups throughout the piece. As demonstrated in [1], if there is no recognisable ground laid, no expectation is produced in the listener and the music is just noise. Likewise if every expected event is granted, the music is predictable and boring. It is the manipulation of this expectation that allows us to build up tension.

Bharucha [2] presented the undirected, spreading-activation neural network of fig. 1 as a model of how *listeners* of CPP music infer structural associations between keys by using only the triad chord as foundation. Given a group of simultaneously sounded tones, activation around the network stabilised around related keys that would have been suggested by traditional music theory. Further studies established that people listening to a group of tones do indeed ‘prime’ related chords in a similar fashion to the network, suggesting expectation [3]. Note that, although based around the triad, the network does indeed capture the relationship between modified chords through how far they are connected. We will here consider Bharucha’s past converged network states as specifying a transition prior for the forthcoming key state.

Bharucha’s network uses a hierarchical structure of keys, chords and tones, with tones linked to chords they occur in and chords linked to keys they occur in (see fig. 1). At each time step, nodes corresponding to groups of the latest ‘heard’ tones are activated, then activation spreads through the network in update cycles until an equilibrium is reached. For example, if the tones of a C Major chord {C, E, G} are ‘heard’ together, the key tones reach equilibrium with energy levels

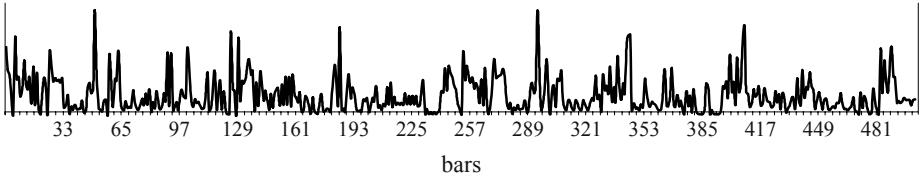


Fig. 2. Graph showing the value of V given in (3) over the entire first movement of Beethoven's fifth symphony

monotonically decreasing as we move away in fifths from C.¹ The converged state of the network nodes, when appropriately normalised, may then be considered to represent a prediction on the next time step's key, chords and notes.

We define *key state* X_t at time step t as the normalised activation vector over keys k once the network is stable, and this is interpreted as a probability density function over the current key. Key state alone is used for making and testing predictions. During convergence, the activation x_j of node j is updated via weights w_{ij} as follows:

$$x_j = \sum_{i:i \rightarrow j} w_{i,j} x_i \quad (1)$$

where $\{i : i \rightarrow j\}$ are the nodes connected to j . $w_{i,j}$ are as in [2]: 0.244 between major chords and keys, 0.22 between minor chords and keys and 0.0122 between tones and chords.

In order to use this network as a model of harmonic expectation the analyser divides a score into temporal units and works through the piece in order, activating the nodes of the tones in each unit as it is heard, and allowing the network to stabilise at each stage. At time t we consider the key state X_t as our observation. We produce a prediction E_{t+1} of the next state by smoothing over all previous observations. We wish predictions to decay by a factor of γ over each beat if not reinforced giving:

$$E_{t+1} \propto \gamma^{1/n} E_t + X_t \quad (2)$$

where we have n units of time per beat. This is a two-variable, first-order Markov process (and is similar to the energy form of a Hidden Markov Model update).

This notion of expectation is meant at a physiological level rather than as a result of logical reasoning and is equivalent to cognitive 'priming' in [3]. We gain a measure of how far our harmonic expectations were violated by considering the KL-divergence of our prediction and observation of the key state X_t :

$$V(t) = \sum_k X_t(k) \log \frac{X_t(k)}{E_t(k)} \quad (3)$$

¹ Whilst we talk of notes in terms of absolute pitch rather than harmonic function (e.g. tonic, dominant), the network's symmetry makes these concepts equivalent within a fixed key.

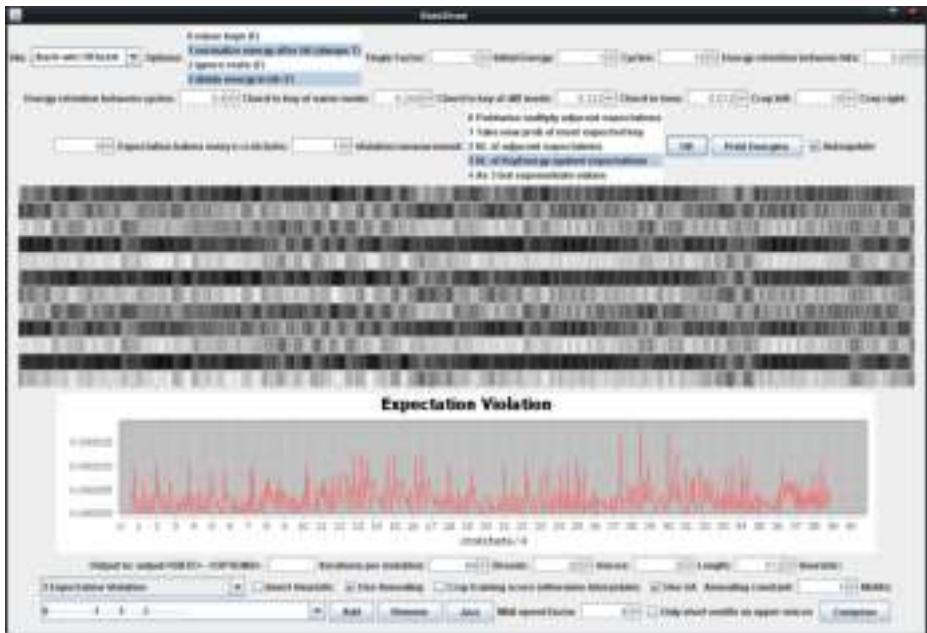


Fig. 3. The interface of the fragment arranger. The central graphic displays the normalised activation vector across the 12 key nodes after the network has stabilised after each unit of time. White is the maximum value and black the minimum.

As t increases, the model's expectation decays if not reinforced so it is not possible to maintain high tension with random notes. Fig. 2 is an example of V over time.

We now use this *local* measure of violation to construct a *global* fitness function R by considering its temporal structure. R compares the temporal tension profile of a candidate score c with that of a training score s :

$$R(c) = \left(\sum_t |V_s(t) - V_c(t)| \right)^{-1} \quad (4)$$

R thus assigns high fitness to new compositions having similar tension profiles to a known training composition, chosen for its pleasing tension profile.

3 The Fragment Arranging Task

As our performance measure is mostly concerned with harmony and less so with rhythm or melodic development we decided to reduce the problem of composing into the easier fragment arranging task. In this task, the program is presented with a number of pre-written motifs and is required to arrange them into a score in a musically desirable fashion. (More advanced motif-arranging systems may be found in [10] and the work of Cope [22].) A motif is defined as a sequence of notes and



Fig. 4. Fragments used in the evaluation, taken from Fugue 4, Book I of Bach's *Well-Tempered Clavier*

is typically a fragment of a melody that might be used several times by a composer. Arranging motifs into a score means deciding when and at what starting pitch they should be played (where motifs may be used any number of times, and may begin at any multiple of halfbeats through a measure). A score is made up of a number of voices and a voice may only be playing one motif at any one time. Far from being a ‘toy problem’ this demands an ingenious approach to harmony and produces interesting results quite different from the training material.

The task is therefore a state-space searching problem where we can move about the space by adding or removing motifs from the score. We implemented evolutionary search using a Metropolis-Hastings (MH) algorithm with simulated annealing [19]. In the MH algorithm the state space is randomly sampled for a proposed alternative to the current state (in our case sampling only involves adding or removing motifs). We accept any proposal that is better than the current state. We also accept some that are worse probabilistically based on how much worse and our *annealing schedule*, which reduces the chance of acceptance as the algorithm progresses. The probability of proposal p being accepted over the current score s was:

$$\Pr(\text{accept}) = e^{100(R'(p) - R'(s)).\beta} \quad (5)$$

where β decreases linearly from 1 to 0 over the course of the algorithm. The fitness function $R' = RS$ used eqn. (4) together with a density evaluator S , which penalised scores with more than three quarters or less than a quarter of their units as rests. As the fitness function does not differentiate between notes of the same tone in different octaves, the octave that each fragment was placed in was decided by hand afterwards. (The MH algorithm may be viewed as a particular case of Genetic Algorithms: maintaining a population of one; using mutation but no crossover; and always generating two children, one of which is identical to the parent [11].)

4 Evaluation

Although the potential for computers in composition extends well beyond imitating human composers, the task of composing lends itself naturally to a Turing Test. The nature of the fragments task is reminiscent of a fugue so we decided for the evaluation to extract four short (2-8 measure) motifs from a Bach fugue



(a) Computer



(b) Human

Fig. 5. Excerpts from the evaluation scores. The full performances may be listened to at www.elec.qmul.ac.uk/digitalmusic/papers/2009/MurrayBrowne09evo-data.

(see fig. 4). The output piece was specified to be in 4/4 time, 32 measures long and in three voices for piano.

We also wished to demonstrate that the performance measure is quite abstracted from the notes and details of the training score, so we are actually modelling the listener's emotional response rather than comparing how similar the candidate score is to our training score. To do so, the same fugue that provided the fragments was used as the training score, to see if the arranger would try to reproduce it.

An experienced composer, Zac Gvirtzman, was given two days to write a piece to the same restrictions with the same input, having heard no program output. Both pieces were recorded by a pianist. We randomly approached 22 Oxford music students in their faculty, told them the composition brief, played them both pieces and then asked them to decide which was written by a human and which by computer. Presentation order was alternated. Excerpts from the two scores are shown in fig. 5. Having heard both pieces, nine answered correctly, 12 answered incorrectly and one refused to answer. Assuming a flat prior on the population's probability of making a correct decision, the observations yield a Beta posterior having mean 0.42 and standard deviation 0.10, showing the human and machine to be effectively indistinguishable.

5 Discussion

The fragment task is by no means trivial, as confirmed by the composer. However, fragment-arranging is just one of many compositional tools that may be

used by humans and larger composing programs, and the human composer commented that he felt very limited without having control over the dynamics and tempo. Even with such a small number of building blocks the program output sounded less repetitive than many Markovian and rule based approaches. Some outputs show original and resolving harmonies and are notable for their tolerance of dissonance that enhance the resolution. Despite not utilising any grammar the outputs have a strong sense of structure, which has been learnt from existing compositions. Interestingly, both computer and human scores used in the evaluation followed an ABA structure, with A drawing exclusively on fragment **A** (see fig. 4). By deliberately violating the listener's harmonic expectation in a controlled way they build up and release global tension, in a similar way to Abdallah and Plumbley's analysis of expectation in notes. They do still however lack a sense of purpose, which will be in part due to the model not having any learning capabilities. Future work may expand on the expectation provided by the listening model by incorporating it alongside Markov chains applied to melody, harmonic progressions and rhythm.

The abstraction of the fitness function from the training material is demonstrated by how, even with both the training material and the set of motifs to be arranged sourced from the same fugue, the output has not attempted to imitate it in any recognisable sense. Further work is required to determine how the function V given in (3) may be used without any training material. Experimenting with the program showed that obvious ways of analysing V such as considering its average over a large corpus of material or attempting to maintain some proportional cumulative value (e.g. $\sum_{i=1}^t V_c(i) = \lambda t$ for some λ) does not produce good results. Averaging V over a large corpus of work such as the *Well-Tempered Clavier* results in a fairly flat graph, dissimilar to any of the individual pieces in the work. This was the reason we used just a single composition for our fitness function rather than a set.

The space of scores searched by our evolutionary algorithm were intensionally constrained, and we do not claim to have constructed a full automated composition system. However the performance of the expectation-violation fitness function on the constrained task is encouraging, and suggests that it could be used to enhance most existing composition systems, if weighted appropriately and incorporated into their fitness functions.

References

1. Abdallah, S., Plumbley, M.: Information dynamics and the perception of temporal structure in music. *Connection Science Journal* (in press)
2. Bharucha, J.: MUSACT: A Connectionist Model of Musical Harmony. In: Proc. Ninth Annual Meeting of the Cognitive Science Society, Hillsdale, New Jersey (1989)
3. Bharucha, J., Bigand, E., Tillmann, B.: Implicit Learning of Tonality: A Self-Organising Approach. *Psychological Review* (2000)
4. Fox, C.W.: Genetic Hierarchical Music Structures. In: Proceedings of the International Florida Artificial Research Society Conference (2006)

5. Gill, S.: A Technique for the Composition of Music in a Computer. *The Computer Journal* (1963)
6. Hiller, L., Isaacson, L.: Musical Composition with a High-Speed Digital Computer. *J. Audio Eng. Soc.* (1958)
7. Hoover, A.K., Rosario, M.P., Stanley, K.O.: Scaffolding for Interactively Evolving Novel Drum Tracks for Existing Songs. In: *Proceedings of the Sixth European Workshop on Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART)*. Springer, Heidelberg (2008)
8. Husbands, P., Copley, P., Eldridge, A., Mandelis, J.: An Introduction to Evolutionary Computing for Musicians. In: *Evolutionary Computer Music*, pp. 1–27. Springer, Heidelberg (2007)
9. Keller, R.M., Morrison, D.R.: A Grammatical Approach to Automatic Improvisation. In: *Proceedings of the Fourth Sound and Music Conference* (2007)
10. Keller, R., Hunt, M., Jones, S., Morrison, D., Wolin, A., Gomez, S.: Blues for Gary: Design Abstractions for a Jazz Improvisation Assistant. *Electronic Notes in Theoretical Computer Science* 193, 47–60 (2007)
11. Laskey, K., Myers, J.: Population Markov Chain Monte Carlo. In: *Machine Learning* (2003)
12. Lo, M.Y., Lucas, S.M.: Evolving Musical Sequences with N-Gram Based Trainable Fitness Functions. In: *IEEE Congress on Evolutionary Computation* (2006)
13. McCormack, J.: Open problems in evolutionary music and art. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2005. LNCS*, vol. 3449, pp. 428–436. Springer, Heidelberg (2005)
14. Meyer, L.: *Emotion and Meaning in Music*. University of Chicago Press (1956)
15. Minsky, M.: *Music, Mind and Meaning*. CMJ (1981)
16. Moorer, J.: Music and Computer Composition. *Communications of the ACM* (1972)
17. Papadopoulos, G., Wiggins, G.: AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects. In: *AISB Symp. Musical Creativity* (1999)
18. Pearce, M., Wiggins, G.: Aspects of a Cognitive Theory of Creativity in Musical Composition. In: *Proceedings of the ECAI 2002 Workshop on Creative Systems* (2002)
19. Rao, R.P.: Bayesian computation in recurrent neural circuits. *Neural Computation* (2004)
20. Schwanauer, S., Levitt, D.: *Machine Models of Music*. MIT Press, Cambridge (1993)
21. Sundberg, J., Lindblom, B.: Generative Theories in Language and Music Description. *Cognition* (1976)
22. Wiggins, G.: Review of Computer Models of Musical Creativity by David Cope. *Literary and Linguistic Computing* (2007)

Composing Using Heterogeneous Cellular Automata

Somnuk Phon-Amnuaisuk

Music Informatics Research Group, Multimedia University,
Jln Multimedia, 63100 Cyberjaya, Selangor Darul Ehsan, Malaysia
`somnuk.amnuaisuk@mmu.edu.my`

Abstract. Music composition is a highly intelligent activity. Composers exploit a large number of possible patterns and creatively compose a new piece of music by weaving various patterns together in a musically intelligent manner. Many researchers have investigated algorithmic compositions and realised the limitations of knowledge elicitation and knowledge exploitation in a given representation/computation paradigm. This paper discusses the applications of *heterogeneous cellular automata (hetCA)* in generating chorale melodies and Bach chorales harmonisation. We explore the machine learning approach in learning rewrite-rules of cellular automata. Rewrite-rules are learned from music examples using a time-delay neural network. After the hetCA has successfully learned musical patterns from examples, new compositions are generated from the hetCA model.

Keywords: Composing by heterogeneous cellular automata, Dynamic neural networks, Automatic music generation, Chorales.

1 Background

It has been demonstrated by previous works that for a system to generate pleasant tonal music, it must possess sophisticated musical knowledge [4], [5], [6], [7]. This knowledge is best captured by experts before being transformed and represented in computers. Two main challenges in building an intelligent system are (i) how is the knowledge obtained? and (ii) how is the knowledge exploited? Although expert knowledge is generally more effective, it is difficult to elicit knowledge from experts. It is also difficult to exploit this knowledge since it always takes the shape of rules. Moreover, search space is exponentially increased as the number of rules increase. Recently, progress in machine learning techniques offers alternative approaches that could answer the challenges rising from knowledge elicitation and exploitation issues.

In this paper, we investigate the *cellular automata (CA)* music composition system in which CA is used to represent musical knowledge and processes for predicting a new pitch for a given context. In CA paradigm, pitches are CA cells where the presence and absence of pitches across time form a piano roll pattern. Instead of explicitly seeing music in traditional structures of form, melody,

harmony and texture, the music is seen as a time series pattern of a non-empty pitch set in which the behaviour of pitches are governed by both local and global dependency among pitches. In this view, traditional musical structures are implicitly captured in CA formations.

In our approach, the behaviour of a CA is learned from given examples where the learned CA model is then used to generate a new piece of music. In this report, we organise the presentation into the following topics: 1. Background, 2. Cellular automata, 3. Our approach 4. Experiments & Discussion, and 5. Conclusion.

2 Cellular Automata

Cellular automata (CA) are dynamic systems with discrete space and time. Space is represented as an array of cells and time is perceived as a discrete time step. Each CA cell may have two or more discrete states. CA may have more than one dimensions. We could always include a temporal information to CAs. For example, a one-dimension CA could be represented as a two-dimension array where each row represents space and each column represent time.

Composing Using Cellular Automata. Although algorithmic music composition with cellular automata has been investigated before by previous works such as [12], [11] and [17] (see more review in [2]). Previous works in algorithmic music composition look at CA compositions from an experimental music perspective (i.e., focus on interestingness of generated musical pieces using CA paradigm). This work, however, investigates CA from a different perspective. We are more interested in *the issue of whether CA would be able to learn musical examples, recall the examples and generate new compositions in the style of what it has learned*. In this work, we investigate the use of heterogeneous CA to learn and generate chorales.

Cellular automata (CA) are called homogeneous CA if all the cells in the CA share the same rewrite-rules. In heterogeneous CA, different cells have different rewrite rules. CA can be used to represent a piano roll in this fashion. Let $a_i(t)$ denotes a cell of a CA where $i \in \{C2, C\#2, \dots, B6\}$. To simplify the notation, let us think of i as integers from 1 to 60 where integer 1 denotes C2, 2 denotes C#2 and so on. A vector $a_{i-r}(t), a_{i-r+1}(t), \dots, a_i(t), \dots, a_{i+r}(t)$ is a single dimension CA in which the prediction of $a_i(t+1)$ is governed by the neighboring cells of $a_i(t)$.

$$a_i(t+1) = f(a_{i-r}(t), a_{i-r+1}(t), \dots, a_i(t), \dots, a_{i+r}(t)) \quad (1)$$

If a_i is a binary state cell where the two states represent a note-on and a note-off state, then a vector $a_{i-r}(t), \dots, a_{i+r}(t)$ could represent 2^{2r+1} rewrite rules. In our representation, the note duration is the amount of time. Here, we fix a discrete time step t at a duration equivalent to a quaver note.

Generating a new note by rewrite-rules could be either deterministic or non-deterministic. In this paper, we employ a neural network to learn the rewrite-rules from given examples. The learned model is later used to generate (i.e., predict)

a new note. The prediction of a new note in this manner is deterministic (see equation 1) and it is local in nature. It also does not exploit any historical information. We shall discuss how to include historical information (e.g., memory) into the model next.

Exploiting Memory in a New Note Prediction. Mozer's *CONCERT* system is an example of systems which compose by prediction [13]. He explores the benefits of psychoacoustic representation in his CONCERT system which composes music using recurrent auto-predictive neural network. CONCERT takes in one note and predicts the next note. The representation of input notes is psychoacoustic-based and the output notes carry the following information: pitch, duration and chord. Temporal effect is represented in the context layer which is a recurrent structure. In brief, the CONCERT system exploits historical events n_{t-n}, \dots, n_t to predict the next note n_{t+1} . Similar concepts are also being explored earlier by [3] and recently by [14]. In the same fashion, the prediction of the next note at CA_{t+1} could be conditioned by the context of input from CA_t, \dots, CA_{t-n} . We can modify equation 1 above to include historical context by:

$$a_i(t+1) = f(a_{i-r}(t), \dots, a_{i+r}(t), a_{i-r}(t-1), \dots, a_{i+r}(t-n)) \quad (2)$$

3 Our Approach

3.1 Representing CA as a Piano Roll

In our approach, we explore variations of representation from a CA perspective. Instead of representing the melody line as a sequence of notes, our input-output pair is a sequence of \langle set of input notes, set of output notes \rangle . In Figure 1 above, there are 3 input notes and one output note. Figure 1 (a) and (b) shows two main representation styles of the given *Ode to Joy* theme. With this simple introduction, readers may already see that there are many variations of this representation scheme and this offers various ways to capture different context. Representation in the style of Figure 1 (a) captures local melodic context while representation in the style of Figure 1 (b) captures local harmonic context.

3.2 Learning CA Rewrite Rules Using Neural Networks

A multilayer perceptron (MLP) is a popular neural network structure and it has been widely employed as multi-class classifiers. MLP could be used to capture input-output patterns in Figure 1. Depending on the number of pitches in the input set (i.e., three in the example in Figure 1), binary input of length three would be able to classify up to 8 patterns. That is, we can have up to 2^n patterns for the input of length n . Unfortunately, due to the complexity of the music domain, ambiguity always exists even when the input length is as high as 88 notes or 2^{88} patterns. Instead of increasing the number of input notes, we employ two tactics (i) employing heterogeneous CA and (ii) exploiting temporal context by using time-delay neural networks.

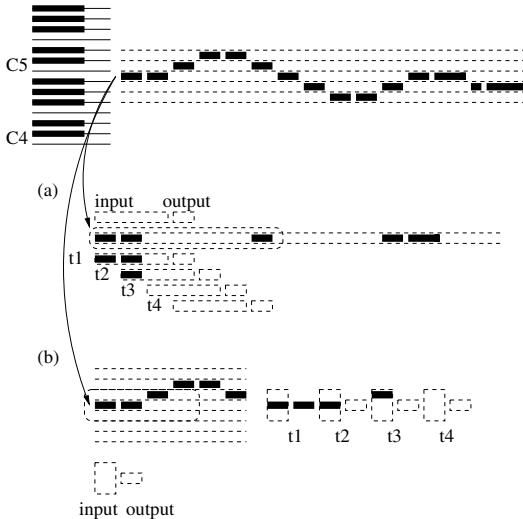


Fig. 1. The first 15 notes from the Ode to Joy theme and examples of two encoding styles which emphasize (a) melodic context and (b) harmonic context

Employing Heterogeneous CA. In this experiment, we have decided to represent input/output of the neural network in the style of Figure 1 b. Pitches between C2 to B6 are allowed in this experiment, that is five octaves or 60 pitches. The duration of each input note is fixed at a quaver. For each pitch in the piano roll (e.g., C2), the rewrite rules are learned independently (i.e., each pitch has its own model). Hence, this setup realises a heterogeneous CA.

Employing Temporal Contexts. Dynamic networks are generally more powerful than static networks since temporal contexts are added into the model via delayed input or recurrent input. Here we report our experimental results from the *time-delay neural network (TDNN)* which is a delayed input neural network.

Let $u(t) \in \{0, 1\}$ and $y(t) \in R$ be the input and output at time t , we can express the relationship between output y and input u of the static (i.e., multi-layer feed-forward network) and dynamic networks (i.e., TDNN) in equations 3 and 4 respectively:

$$y(t) = f(u(t)) \quad (3)$$

$$y(t) = f(u(t), u(t-1), u(t-2), \dots, u(t-m)) \quad (4)$$

The time-delay MLP exploits delayed input while the standard MLP is static and does not exploit any temporal context. Figure 2 (b) shows the architecture of the TDNN network employed in this experiment. All experiments reported here were carried out with the following parameters: neighborhood size of CA is 3 (e.g., $r = 3$), time-delay is 7 steps and the network has 56 input nodes, 64 hidden nodes and 1 output nodes. Figure 2 (c) gives an overview of the system.

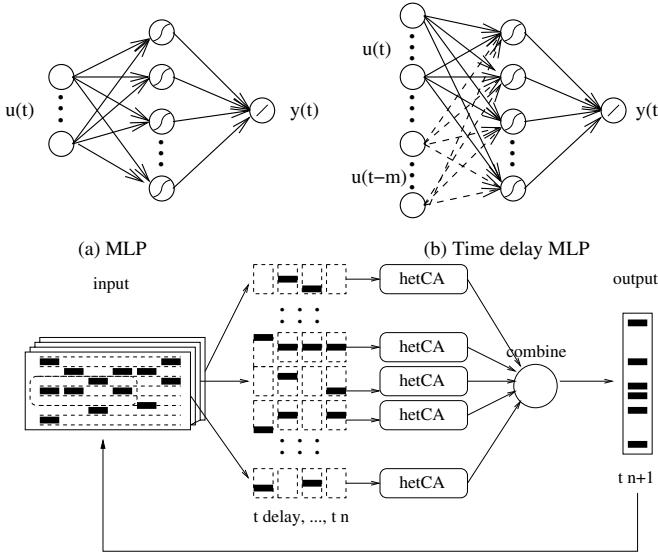


Fig. 2. Network architecture of (a) MLP (b) Time-delay MLP and (c) Overview of the system

The original input is partitioned into many sets of $2^{r+1} \times T$ sequences, each will be used to train a CA. In the composition phrase, a seed (i.e., a few CA time steps) is given to initialise the composition process. The generated output then become the input of the next time step (see Figure 2) and the composition is terminated when the desired time step is reached.

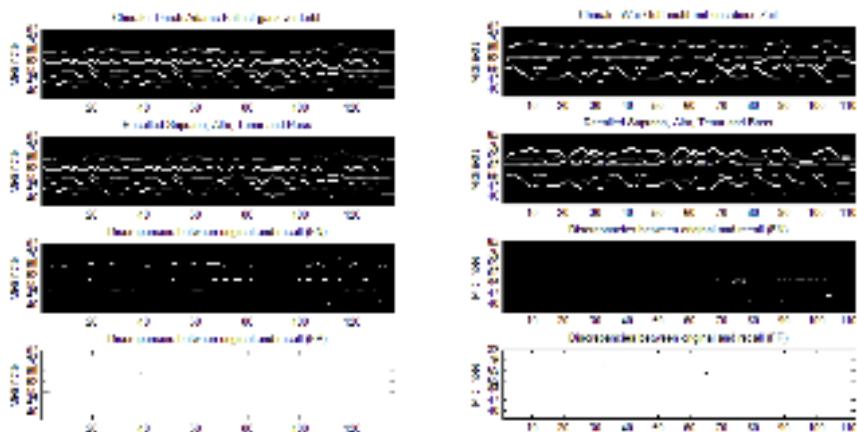
Therefore, in our architecture, TDNN exploits both harmonic context and melodic context (please refer to Figure 1). Therefore, TDNN could learn CA rewrite-rules which capture the local melodic, harmonic and textual contexts of a given piece.

4 Experiments and Discussion

We have selected four pieces from *Riemenschneider 371 harmonised chorales and 69 chorale melodies* [16] for evaluation. The fact that music is a subjective topic and its evaluation involves both subjective and objective dimensions, to evaluate our proposed approach, we devise two experiments for two main objectives. For the first objective, we investigate the precision and recall rate of our hetCA models. This verifies that the model could successfully learn the signature of a given piece. For the second objective, we investigate whether the model could be used to generate a new piece. The performance of the first objective can be evaluated objectively. However, the performance of the second objective can only be evaluated objectively up to the syntax level (i.e., whether the generated composition has violated the normal part writing practice, but the quality of the composition is a subjective matter and will not be addressed in the present

Table 1. Evaluate precision and recall performances

Training example	TP	FP	FN	Recall	Precision	f	Figure
Melody (Soprano)							
Durch Adams Fall ist ganz verderbt	125	2	11	91.91	98.43	0.95	-
Wär' Gott nicht mit uns diese Zeit	111	2	1	99.11	98.23	0.98	-
Her Jesu Christ, du höchstes Gut	115	3	5	95.83	97.46	0.96	-
Wir Christenleut'	85	1	3	96.59	98.84	0.97	-
Harmony (SATB)							
Durch Adams Fall ist ganz verderbt	480	2	64	88.24	99.59	0.93	3, 4, 5, 6
Wär' Gott nicht mit uns diese Zeit	429	1	19	95.76	99.77	0.97	3, 5
Her Jesu Christ, du höchstes Gut	469	2	11	97.11	99.58	0.98	-
Wir Christenleut'	338	1	14	95.76	99.71	0.97	-

**Fig. 3.** Piano rolls of the original chorale, recalled, TP and FP. On the left 'Durch Adams Fall ist ganz verderbt' and on the right 'Wär' Gott nicht mit uns diese Zeit'.

study). The performance of hetCA learned by TDNN were objectively evaluated using *recall* and *precision* measurements defined below:

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

$$f = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

4.1 Precision and Recall Rate of hetCA

Four Bach Chorales (Riemenschneider ID 100, 182, 266 and 321) were chosen in this experiment. Eight hetCA models were built, four models for four soprano melodies and the other four models for chorale harmonisation. The top block of

table 4 shows the precision and recall rates of the soprano melody alone while the bottom block shows the precision and recall rates of the chorale harmonisations (i.e., soprano, alto, tenor and bass). The recall and precision are quite good for both melody and harmony recall tasks. To our knowledge, this approach is novel and we could not directly compare the recall and precision rates of this work with other related works.

Figure 3 illustrates the behaviours of two hetCA models (these were typical output). The first row is a piano roll of original harmonisation. The second row is the recalled harmonisation (i.e., the system predict notes at time $t + 1$ from input at time t). The third and the forth rows are FP and FN . Each white dash unit is counted as one FP (see 3rd row) and each black dash unit is counted as one FN (see 4th row). From the figures, FP and FN are calculated from the difference between original chorales and reconstructed chorales: $FP = \max(\text{Reconstructed} - \text{Original}, 0)$ and $FN = \max(\text{Original} - \text{Reconstructed}, 0)$ where $\max(a, b)$ returns the a if $a \geq b$ otherwise returns b . The result shows that the models have successfully learned a Bach chorales' harmonisation patterns. The recalled harmonisation was also transcribed into a standard score format for readers' convenience (see Figure 4).



Fig. 4. Recall of a chorale “Durch Adams Fall ist ganz verderbt”

4.2 Composing a New Chorale Piece

After learning, the model was also used to generate a new harmonisation. The result is shown in Figure 5. The piano roll of the generated output from the left hand side is transcribed to a traditional score in Figure 6.

It is obvious that although the system was capable of learning patterns presented to the system and reproducing what it has learned effectively (i.e., evidences from high recall and precision rates), the system could not reproduce a convincing chorale harmonisation as compared to previous works in this area [6], [9], [15], [1]. Major weaknesses could be pointed out as below:

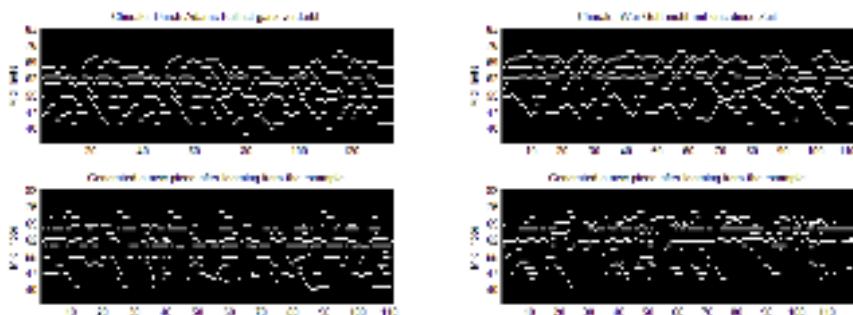


Fig. 5. Piano rolls of the training example and the new composition



Fig. 6. A composition from a hetCA after training

1. Voice leadings are stylistically incorrect,
2. harmonic movements are stylistically incorrect and
3. the music has no direction, no intention and no global structure.

This is a serious problem and it can be corrected if we incorporate appropriate knowledge into the system. There are various alternatives for this deficit¹. However, we do not want to handcraft the knowledge explicitly since our goal is to let the system learn this knowledge from CA pattern formations.

After being critical, we would like to point out that this does not mean that there is no hope with this approach. On the contrary, we believe there is a rich area to be explored here. The above composition is, by the way, not bad at all,

¹ Note that [6] and [15] have their knowledge-bases explicitly handcrafted; [9] and [1] have handcrafted knowledge of *functional harmony* in their melody-chord realisation parts and in the harmonisation processes (i.e., the division of the whole process into melody-functional harmony, voices and ornamentations sub-processes), [9] also has explicit handcrafted knowledge in the ornamentation part of the system.

if we take into account that no explicit knowledge is coded in the system at all. All knowledge in the system is purely from the formation of the training data, the simplicity (which does not mean weak) of the process is very appealing. It would be hard to generate this kind of output using other techniques. More random notes should be expected from output generated using other non-knowledge intensive techniques.

5 Conclusion

If we could create a look up table for all songs that have ever been composed so far, then it is imaginable that we would be able to predict the next note of any given sequence with absolute 100% recall rate, provided that we can access all of its context. This humongous look up table may be extended to be a composition system that could generate new musically sensible compositions using existing materials. However, this is far from being practical. Therefore the aim of model learning should be to capture important stylistic contexts (i.e., both local and global stylistic contexts) into the model, so that new compositions could be generated from the model. To capture an appropriate stylistic context using CA pattern formations, the representation schemes of CA must be devised to capture appropriate melodic, harmonic, textual and formal contexts of training example.

In this report, we present a composition system that relies on cellular automata. We have shown examples of CA compositions generated from heterogeneous CA. Rewrite-rules of a CA is learned using time-delay neural network. We evaluate the proposed approach objectively and the results shows that CA could successfully learn complex patterns. In further work, we shall investigate the effectiveness of the approach in learning stylistic information. Hybrid systems between other machine learning approaches and CA would be another area which could be interesting to further explore them.

Acknowledgement. I would like to thank the reviewers for their useful comments.

References

1. Allan, M., Williams, C.K.: Harmonising chorales by probabilistic inference. In: Saul, K.L., Weiss, Y., Bottou, L. (eds.) *Advances in Neural Information Processing Systems*, vol. 17. MIT Press, Cambridge
2. Burraston, D., Edmons, E.: Cellular automata in generative electronic music and sonic art: Historical and technical review. *Digital Creativity* 16(3), 165–185 (2005)
3. Conklin, D., Witten, I.H.: Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24, 51–73 (1995)
4. Cope, D.: *Computer and Musical Style*. Oxford University Press, Oxford (1991)
5. Cope, D.: *Virtual Music: Computer Synthesis of Musical Style*. MIT Press, Cambridge (2001)

6. Ebcioğlu, K.: An expert system for harmonizing four-part chorales. In: Balaban, M., Ebcioğlu, K., Laske, O. (eds.) *Understanding Music with AI: Perspectives on music cognition*, ch. 12, pp. 294–333. AAAI Press/MIT Press,
7. Fry, C.: Flavors Band: A language for specifying musical style. In: Schwanauer, S.M., Levitt, D.A. (eds.) *Machine Models of Music*, ch. 19, pp. 426–451. MIT Press, Cambridge
8. Gartland-Jones, A., Copley, P.: The suitability of genetic algorithms for musical composition. *Contemporary Music Review* 22(3), 43–55 (2003)
9. Hild, H., Feulner, J., Menzel, W.: HARMONET: A neural net for harmonizing chorales in the style of J.S. Bach. In: Lippmann, R.P., Moody, J.E., Touretzky, D.S. (eds.) *Advances in Neural Information Processing*, vol. 4, pp. 267–274. Morgan Kaufman, San Francisco (1991)
10. Horner, A., Goldberg, D.E.: Genetic algorithms and computer-assisted music composition. In: Belew, R., Booker, L. (eds.) *Proceedings of The Fourth International Conference on Genetic Algorithms*. Morgan Kauffman, San Francisco (1991)
11. McAlpine, K., Miranda, E., Hoggar, S.: Making music with algorithm: A case study. *Computer Music Journal* 23(2), 19–30 (1999)
12. Miranda, E.R.: Cellular automata music: An interdisciplinary project. *Interface* 22(1), 3–21 (1993)
13. Mozer, M.C.: Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. In: Griffith, N., Todd, P.M. (eds.) *Music Networks: Parallel Distributed Perception and Performance*. MIT Press, Bradford Book (1999)
14. Oliwa, T., Wagner, M.: Composing Music with Neural Networks and Probabilistic Finite-State Machines. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008. LNCS*, vol. 4974, pp. 503–508. Springer, Heidelberg (2008)
15. Phon-Amnuaisuk, S.: Control language for harmonisation process. In: Anagnostopoulou, C., Ferrand, M., Smaill, A. (eds.) *ICMAI 2002. LNCS*, vol. 2445, p. 155. Springer, Heidelberg (2002)
16. Riemenschneider, A.: 371 Harmonized Chorales and 69 Chorale Melodies with Figured Bass. G. Schirmer, Inc. (1941)
17. WolframTones, <http://tones.wolfram.com>

On the Socialization of Evolutionary Art

Juan Romero¹, Penousal Machado², and Antonino Santos¹

¹ Faculty of Computer Science, University of Coruña, Coruña, Spain

jj@udc.es, nino@udc.es

² CISUC, Department of Informatics Engineering, University of Coimbra,

3030 Coimbra, Portugal

machado@dei.uc.pt

Abstract. The lack of a social context is a drawback in current Interactive Evolutionary Computation systems. In application areas where cultural characteristics are particularly important, such as visual arts and music, this problem becomes more pressing. To address this issue, we analyze variants of the traditional Interactive Evolutionary Art approach – such as multi-user, parallel and partially interactive approaches – and present an extension of the traditional Interactive Evolutionary Computation paradigm. This extension incorporates users and systems in a Hybrid Society model, that allows the interaction between multiple users and systems, establishing $n - m$ relations among them, and promotes cooperation.

1 Introduction

Interactive Evolutionary Computation (IEC) is a variation of Evolutionary Computation in which the fitness of the individuals is determined, at least to some extent, by a subjective evaluation carried out by a human user. In recent years, this paradigm has been applied to various fields. Some of these domains possess a high social component, for instance, those related to aesthetics such as art, music, design, architecture, fashion, etc [1,2,3]. We will call these domains *social domains*, and these systems, *interactive evolutionary art (IEA) systems*.

In social domains we may distinguish two roles: the *creator* and the *critic* or audience. The canonical IEA system integrates only two participants: an Evolutionary Computation (EC) system and a human user. The role of the creator is played by the EC system, while the role of the critic is played by the human who assigns fitness to the generated *products*. Products stand for any kind of artifacts (ideas, approaches, artworks, information, solutions, etc.). Fig. 1 illustrates the relationships established in canonical systems.

The fitness of the creators depends on the user's taste. Moreover, the user does not always wish to find an “optimal” product. Usually he/she wants one or more products that satisfy his/hers subjective preferences.

Due to their nature, social domains, present a series of characteristics [4] that complicate IEC systems' design:

- The existence of several different fitness functions. Each user assigns fitness according to their individual criteria and taste.



Fig. 1. Canonical IEA system

- The dynamic character of the fitness.
- The difficulty to define fitness in a formal way [5].

Additionally, canonical IEC systems have several shortcomings that hinder their applicability to social domains:

1. They lack a cultural environment, such as the one existing in human societies; “The value of an artwork depends on its surrounding cultural context (or contexts)” [5].
2. The lack of cooperation among users. The results attained by certain users are not shared with other ones.
3. The lack of evaluation abilities. Canonical IEC systems are unable to evaluate their products, which forces the user to evaluate the whole population. As a consequence, population size and length of the evolutionary runs tends to be small.
4. Related to the previous issue, user fatigue caused by the need to evaluate a large number of individuals raises a wide variety of problems.

The aforementioned problems are related with the 1 – 1 relation between user and system established in canonical IEC systems and can be alleviated by establishing an n-m relationship model.

This 1 – 1 relation also makes difficult the integration of IEC in other systems, and the creation of complex systems that incorporates IEC systems. Several authors [1,2,4,3] have remarked the need for fostering common environments which allow the interaction and validation of several IEC systems. A model that allows the integration of multiple IEC systems and users, establishing an $n - m$ relation, can be of great value to the research community providing a way to validate and compare several IEC systems, and promoting the cooperation among researchers by providing a common framework.

We propose a paradigm, called Hybrid Society (HS), which extends the capabilities of current IEC allowing the establishment of $n - m$ relationships between evolutionary computation systems and users, as well as the incorporation of human creators and artificial critics [6]. We start by making an analysis of the existing variations of the IEC paradigm. Afterwards, we make a brief conceptual description of the HS architecture and of its main mechanisms. Finally we draw some overall conclusions.

2 IEC Paradigm

In this section, we analyze the current extensions of the canonical IEA system.

2.1 Multi User

One of the most common variations of IEA is the multi user approach, in which the evaluation of the products is performed by a set of users instead of a single one. Typically (see e.g. [7]) the fitness of each product is determined by the average of the evaluations made by the users. This approach has one severe shortcoming: when the users have different preferences the results will hardly be satisfactory.

In social domains that involve highly subjective criteria this problem is accentuated. The lack of consistency in the evaluations can jeopardize the evolutionary process, due to the existence of contradictory assessments resulting from the different preferences of the users. Considering the average assessment can lead to the evolution of products that, although relatively well sanctioned by the community, are not entirely pleasing to any of the individual users. In addition, it might lead to a “dictatorship of the majority” where the interests and preferences of minority groups are never satisfied.

2.2 Parallel

The parallel IEA variant is characterized by the use of a set of IEA systems whose EC System interchange individuals of the population. Fig. 2 shows the parallel IEA model.

This approach poses a problem in fields where fitness is assigned in accordance with subjective criteria such as individual taste. In this type of domains, it would be necessary to integrate a mechanism that maximizes the migration of products among the IEA systems of users with similar preferences, and that minimizes the transfers among the ones that have dissimilar or opposite tastes.

This approach has another important shortcoming, it is only useful when the IEA systems use the same representation for the individuals. To use different

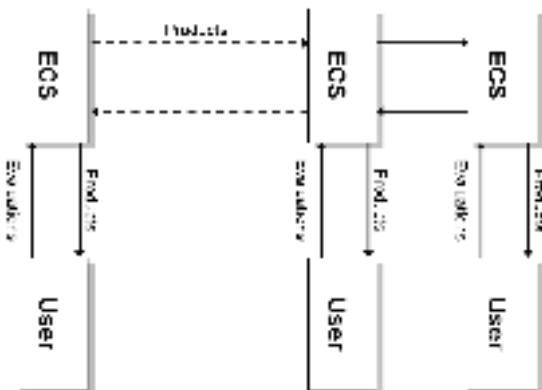


Fig. 2. Parallel IEA

IEA systems, one would need to devise a way of translating the products from one system to the other.

This can become a huge problem, especially if one wishes to incorporate a set of heterogeneous IEA systems. The problem, once again, is linked with the way that preferences and taste are communicated from one system to the other, which, in this case, happens indirectly by the migration of highly fit individuals.

Probably due to the previously mentioned shortcoming, we were unable to find examples of parallel IEA systems applied to social domains.

2.3 Partially Interactive

Another extension of the IEA paradigm consists in the integration of evaluation mechanisms. These can have two different goals: performing some sort of evaluation task that simplifies the job of the user, for instance, eliminating products that are invalid; predicting the evaluations of the user, thus enabling the system to run in standalone mode, even if it is just during short periods of time. In both cases the integration of automatic evaluation mechanisms may contribute to the decrease of user fatigue and to an increase of quality of the overall result¹.

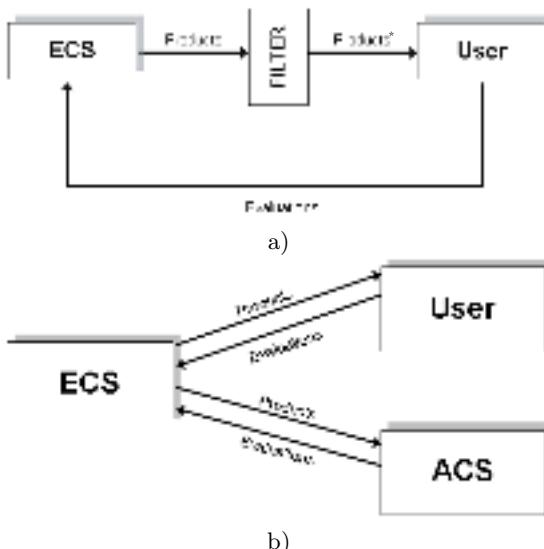


Fig. 3. Partially IEA systems with a filtering layer (a) and an AC (b)

Fig. 3a illustrates the functioning of a partially interactive EC system applied to art, with a filtering layer, while figure 3b presents a system where the evaluations of the products are performed either by an Artificial Critic (AC) or by a human user.

¹ Since less user evaluations are needed longer runs can be performed and consequently better products attained.

In [8] the authors describe a partially interactive evolutionary system, which integrates both components. A filtering layer eliminates images "too simple or too complex, e.g. completely blank or noise (i.e. completely random)" [8]. The AC assigns fitness to the remaining images. The user can interfere in any point of the evolutionary process, by giving its own assessment of the population images, thus overriding the evaluations of the AC.

Another active research line concerns the independent development of ACs. For instance, [9,10,8,11] describe ACs for visual art and music domains, that carry out tasks of author and style identification, attaining accuracy rates

These systems are based on a pre-processing of the artworks, extracting a series of measurements, which are then used as input to Artificial Neural Networks.

The use of these independently developed ACs in the context of IEA may prove interesting and useful, both for increasing performance and for testing purposes. The development of a generic architecture which allows the effortless integration of several ACs and ECs can be of great interest for the development of complex systems, enabling the comparison of different approaches, and promoting the collaboration among research groups.

3 Hybrid Society

Bearing in mind the problems existing in the different IEA variants, this section describes HS, as an extension of the IEC paradigm applied to art domains.

The design of this extension is based on a set of goals:

1. Allowing each user to interact simultaneously with several EC systems.
2. Allowing each EC system to interact with different users.
3. Allowing the participation of generic ACs which evaluate the products created by different EC systems.
4. Providing a way to evaluate the performance of the EC systems in accordance to its capacity to satisfy the preferences of a set of users and of its ability to adapt to changes in these preferences.
5. Allowing a greater degree of interaction between participants with similar preferences, fostering the development of groups with common interests and tastes.
6. Simulating some aspects of the behavior of human society.

HS was designed specifically for social domains and is, therefore, based on a "social conception". According to this view only those products which are found to be interesting by a given *cultural surrounding* are valued. A cultural surrounding may be defined as a set of agents with a high degree of cultural affinity.

This conception does not discard minority trends, as one might infer. If a particular work of art raises the interest of a small community, it will be valued. For instance, Jazz does not have mass appeal. However, in HS the participants that like jazz, and the creators and artificial critics who have adapted to that style, are grouped together in a, possibly thriving, subgroup.

Next we will describe the architecture of HS and its main mechanisms.

3.1 Architecture

The HS architecture consists of a central element called *scenario*, together with a set of participants who communicate with it. Participants may be either artificial or human, and they may play the roles of creator or critic.

Conceptually, a scenario is the common ground of the beings participating in a society. It includes the rules of the game and defines the principles of communication among those beings. Formally, it is the set of applications (databases, communication protocols and interfaces) with which creators, critics and products interact.

In HS the artificial creators are instances of an EC system similar to the one used in standard IEC. The artificial critics are instances of an AC system, similar to those described in section 2.3. Creators, humans or artificial, send products to the scenario. Critics perform evaluations of the products belonging to the scenario, communicating these evaluations by means of bets. The human or artificial nature of a participant is hidden from the rest. Fig.4 shows the relations among the different types of participants and the scenario.

Since there is no longer a direct communication between one user and one EC system, a series of mechanisms must be put in place in order to regulate and monitor the integration of all of them. These mechanisms are: energy exchange, affinity, and offspring generation. The upcoming sections briefly explain these mechanisms, as well as the main variables related to them.

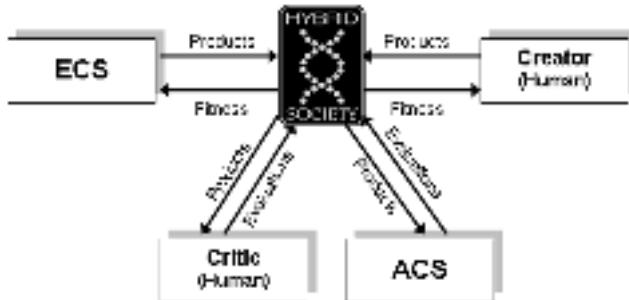


Fig. 4. Hybrid Society Model

3.2 Energy Exchange

The HS energy exchange mechanism makes it possible to determine the adaptation of the participants to the cultural context. Each participant possesses an energy which is, in this context, the measure of its success. This can be seen as an expansion of the fitness concept, in order to evaluate the adaptation of artificial creators and critics.

One of the HS parameters is the initial energy value of each participant. If the energy of a participant becomes less or equal than 0, it is virtually dead and it can no longer participate in the society.

Humans and artificial participants gain and lose energy accordingly to the same rules. Depending on being a creator or a critic, different rules apply.

Each time a creator sends a product to the scenario, a certain amount of energy is subtracted. The creator receives energy when other participants place bets on its products.

The critics can evaluate products and place bets on the creators of the ones they find interesting. A bet is an energy transfer from a critic to a creator. The value of the bet must always be positive and less than the current energy of the critic placing the bet. Once placed, all bets are irrevocable.

When a critic bets on a creator, it gets in return a *possession percentage* of that creator. Possession percentage is defined as the ratio between the value of the bet and the energy of the creator when the bet was placed.

$$\text{possession percentage} = \frac{\text{value of bet}}{\text{energy of creator}} \quad (1)$$

Each time a bet is placed the following actions take place:

1. The value of the bet is subtracted from the energy of the critic placing it.
2. A percentage of the value of the bet is distributed among the previous punters (if there are any) in proportion to their possession percentages. Each of these critics receives energy according to the following formula:

$$\text{profit}_i(t) = \frac{\text{possession percentage}_i}{\sum_{j=1..m} \text{possession percentage}_j} * E(t) * C, \quad (2)$$

where $E(t)$ is the value of the bet placed on the creator at instant t and C an adjustable parameter.

3. The remaining value of the bet is summed to the energy of the creator.

A bet placed on a creator with a small amount of energy can be more profitable to the critic than a bet placed on a creator with a vast amount of energy.

The creators which are valued by the participants of HS obtain great amounts of energy, since they receive many bets. The creators that do not fulfill the demands of the society do not receive energy.

The success of the critics depends on their ability to place bets on interesting products that other members of the society find attractive. If the critic bets on “uninteresting” creators, it will never recover the energy used for the bet. If the critic bets on very “interesting” creators, these creators will later receive a vast number of bets. Since the critic acquired a possession percentage, he will receive part of these bets and thus eventually increase its energy.

This description might lead to think that the best strategy for any critic is that of betting on a popular creator. In practice, the best strategy is to bet on emerging creators, i.e. those creators which aren’t currently valued but who will be in the future.

3.3 Affinity

This mechanism strengthens the relations between participants with high *affinity*. Two participants have high affinity if: they share the same preferences (affinity among critics); when one values the products of the other (affinity among critic and creator); when the products created by them are valued by the same participants (affinity among creators).

When two critics have a high degree of affinity, they will have access to more products which have been positively appraised by the other. For instance, if we have a community where some of the participants like Jazz and others like Rock, the affinity mechanism will foster the establishment of relations between members with similar tastes.

Two mechanisms based on a spatial representation are used in order to establish affinity relations. The first one makes it more likely for critics to receive products which are spatially close to the critic. The second one displaces products, critics and creators according to the evaluations of the products performed by the critics.

The spatial representation used may consist of two, three or more dimensions. Every participant holds a position in this representation. Initially, every participant is located at a random position. When a creator sends a product, this product is placed in a random position in the vicinity of the creator's current position. The maximum allowed initial distance between creator and product is an adjustable parameter.

To foster the establishment of affinity relations, the list of products of each critic is composed by a higher percentage of products within its vicinity than outside its vicinity. These percentages and the maximum distance defining the vicinity zone are adjustable parameters.

Participants and products with high affinity become close according to the critics' evaluations. When a critic issues a positive evaluation of a creator, three movements take place: (i) the critic moves towards the product (ii) the product becomes closer to the critic (iii) the creator of the product moves towards the critic. Each movement is independent and the distance travelled randomly chosen. The maximum distance per movement for creators, critics and products is established by independent adjustable parameters.

Conversely, there are rejection forces – of an opposite direction and lesser magnitude than the previous ones – between participants and products with low affinity.

The use of space fosters the definition of subcultures or subgroups within a scenario, given that the dynamics of HS favors the proximity of participants who have high affinity.

3.4 Offspring Generation

HS is designed so that the participants are able to meet the demands of a dynamic society. This is done following an evolutionary approach.

The offspring generation mechanism is managed by HS. When a participant exceeds a certain energy threshold, HS can create new artificial participants that

are its descendants. When created, the descendent will have half of its parent energy.

By default, the creation of a new descendent is done by the mutation of the genetic code of the progenitor. This genetic information codifies a set of adjustable parameters that allow to change the behavior of the system. An example may prove useful to clarify the previous sentence. Let's consider the creation of an offspring of a creator system. The creator is a typical evolutionary system that creates pieces of art. The genetic code of the creator codifies a set of parameters of the EC algorithm, for instance: mutation rate, crossover probability, population size, etc. By mutating this genetic code, HS creates a new instance of the creator with a different behavior. In the case of critics, the genetic code can, for instance, codify the weights of the different criteria used in the evaluation.

The decision about the exact contents of the genetic code is made by the developers of the original participant. Additionally, the default descendent creation mechanism (mutation of the progenitor) can be overridden by one specifically designed for a particular system.

4 Conclusions

In this paper we presented an extension of the traditional IEC paradigm named Hybrid Society. HS was specifically designed for artistic and social domains, where the evaluation of the products depends on subjective and cultural criteria. To address some of the shortcomings of IEC systems, HS allows the interaction between a multitude of creators and critics, promoting the emergence of clusters of participants with high affinity relations.

This paradigm constitutes an evolution of IEC, providing it with a social component and giving rise to a new level of collaboration between developers and users. These are some of the advantages of our approach:

- There is a common interface shared by several IEC systems.
- The decisions of other users are taken into account in their social dimension. Users (and artificial participants) are located according to their tastes, so that the user's environment is composed of those participants which are aesthetically closer.
- General ACs are considered; these systems can perform evaluations and adapt to the general aesthetics of their vicinity.
- It is domain-independent. The conveyed product is transparent to the scenario, it is only sent between participants.

Given the maturity of the evolutionary art research area and its solid achievements, we consider that this is the correct timing for a common research effort, together with a shift of the relationship paradigm; from a $1 - 1$ relationship between user and IEC system to a $n - m$ relationship. We believe that the use of HS can be an important step in the development of IEC and ACs systems in social and creative domains, fostering collaboration for the creation of IEC systems, and their validation in a dynamic and complex environment.

Acknowledgements. This research is partially funded by the Spanish Ministry for Science and Technology, research project *TIN2008 – 06562/TIN*, “Aplicación de métricas para la clasificación de imágenes según criterios estilísticos y estéticos”. We would like to thank Asis García Chao, Sergio Barreiro, Rosa Reinoso and Mercedes Vara for their help in implementation and experimental work.

References

1. Bentley, P., Corne, D. (eds.): Creative Evolutionary Systems. Morgan Kaufman, San Francisco (2001)
2. Johnson, C., Romero, J.: Genetic algorithms in visual art and music. *Leonardo* 35(2), 175–184 (2002)
3. Todd, P.M., Werner, G.M.: Frankensteinian methods for evolutionary music composition. In: *Musical networks*, pp. 313–339. MIT Press, Cambridge (1999)
4. Pazos, A., Santos, A., Arcay, B., Dorado, J., Romero, J., Rodríguez, J.: An application framework for building evolutionary computer systems in music. *Leonardo* 36(1), 61–64 (2003)
5. Machado, P., Romero, J., Manaris, B., Santos, A., Cardoso, A.: Power to the critics — A framework for the development of artificial art critics. In: IJCAI 2003 Workshop on Creative Systems, Acapulco, Mexico (2003)
6. Romero, J.: Metodología Evolutiva para la Construcción de Modelos Cognitivos Complejos. Exploración de la Creatividad Artificial en Composición Musical. PhD thesis, University of Corunha, Corunha, Spain (2002) (in Spanish)
7. Biles, J.: GenJam Populi: Training an IGA via audience-mediated performance. In: Proceedings of International Computer Music Conference, pp. 347–348. International Computer Music Association (1996)
8. Machado, P., Romero, J., Cardoso, A., Santos, A.: Partially interactive evolutionary artists. *New Generation Computing – Special Issue on Interactive Evolutionary Computation* 23(42), 143–155 (2005)
9. Romero, J., Machado, P., Santos, A., Cardoso, A.: On the development of critics in evolutionary computation artists. In: Raidl, G.R., et al. (eds.) *EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003*. LNCS, vol. 2611, pp. 559–569. Springer, Heidelberg (2003)
10. Machado, P., Romero, J., Santos, A., Cardoso, A., Manaris, B.: Adaptive critics for evolutionary artists. In: Raidl, G.R., et al. (eds.) *EvoWorkshops 2004*. LNCS, vol. 3005, pp. 435–444. Springer, Heidelberg (2004)
11. Manaris, B., Romero, J., Machado, P., Krehbiel, D., Hirzel, T., Pharr, W., Davis, R.: Zipf’s law, music classification and aesthetics. *Computer Music Journal* 29(1), 55–69 (2005)

An Evolutionary Music Composer Algorithm for Bass Harmonization

Roberto De Prisco and Rocco Zaccagnino

Università di Salerno
Dipartimento di Informatica ed Applicazioni
84081 Baronissi (SA), Italy
`{robdep,zaccagnino}@dia.unisa.it`

Abstract. In this paper we present an automatic Evolutionary Music Composer algorithm and a preliminary prototype software that implements it. The specific music composition problem that we consider is the so called unfigured (or figured) bass problem: a bass line is given (sometimes with information about the chords to use) and the automatic composer has to write other 3 voices to have a complete 4-voice piece of music. By automatic we mean that there must be no human intervention in the composing process. We use a genetic algorithm to tackle the figured bass problem and an ad-hoc algorithm to transform an unfigured bass to a figured bass. In this paper we focus on the genetic algorithm.

1 Introduction

Computer music encompasses many aspects across computer science and music. In this paper we are concerned with the aspect of *algorithmic composition*, that is we focus on the problem of composing music by means of a computer program, without *any* human intervention.

In particular we are concerned with the *unfigured bass* harmonization problem. In this problem we are given a bass line as input and the goal is to compose other 3 voices to make a complete 4-voice piece of music. The 4 voices are usually called *bass*, *tenor*, *alto* and *soprano*.

We use a two-stage approach: in the first stage we use an ad-hoc algorithm, based on a graph representation of the bass line, to find a suitable “figuration” by using weights to prefer some chords instead of others. In the second stage we use a genetic algorithm to choose the exact position for the tenor, alto and soprano voices. Due to space constraints, in this paper we will describe only the genetic algorithm and we defer the description of the figuration algorithm to an extended version of this paper.

Preliminary results show that our algorithm is able to produce reasonably “good” solutions. It should be noted that a general evaluation metric does not exists, and the goodness of an harmonization is often also a subjective opinion. So we don’t have a mathematical means to compare our algorithmic compositions to other similar work. In the related work section we discuss similarities and differences of our algorithm with other algorithms that consider the same (or similar) problem.

Background. We assume that the reader is already familiar with both music concepts (chords, harmony rules, etc.) and with genetic algorithms.

2 Related Work

Writing automatic composers for the 4-voice harmonization problem has been already considered in several papers. For example Ebcio glu [4] and Schottstedt [12] describe automatic composers based on rules and expert-systems. Another system capable of composing 4-voice chorales (and not only) is the EMI system by Cope [2,3]; the EMI system uses a combination of various techniques (formal grammars, rules, music analysis, pattern matching). Lehmann [8] uses an approach based on neural networks.

None of the above approaches uses genetic algorithms. Among the works on automatic composers that use genetic algorithms we cite the work by Horner and Goldberg [6] for thematic bridging, the work by Biles [1] for Jazz solos, the work by Jacob [7] which use an interactive genetic algorithm (i.e., it needs human intervention). Horner and Ayers [5] use genetic algorithms to harmonize chords progressions.

The works that are closer to our paper are those by McIntyre [9] and by Wiggins, Papadopoulos, Phon-Amnuaisuk and Tuson [11]. In these two papers, automatic composers that use a genetic algorithm are presented. The specific composition problem that they consider is the harmonization of a given melody line.

The work by Phon-Amnuaisuk and Wiggins [10] provides a comparison between the genetic algorithm approach and the rule-based approach. Their conclusion is that the rule-based system delivers a much better output, however the quality of the output, in any system, is basically dependent on the overall knowledge that the system has.

We present a genetic algorithm that, for many aspects, is similar to the ones presented in [9] and [11]. There are however several differences. The first and most important, is that we consider a different problem (although closely related): we aim at harmonizing a given bass line, while the cited previous works harmonize a given melody line. Moreover there are also several differences in the technical aspects of the genetic algorithm used. In particular our crossover operator is fundamentally different from the ones used in those works; their crossover operator does not use music knowledge to produce new chromosomes, while our crossover does make use of harmonic and melodic considerations to produce better individuals. In this sense we are augmenting the genetic approach with music knowledge following the conclusion of [10].

3 Evolutionary Music Composer (EMC)

In this section we present our EMC algorithm. The EMC algorithm takes as input a figured bass line and produces a complete 4-voice harmonization of the bass line.

3.1 Chromosome and Gene Representation

Figure 1 (next section) shows an unfigured bass line (part *a*) and a possible harmonization (part *b*). An harmonization of the given bass line is a *chromosome*; notice that all chromosomes have the same bass figuration that we find with the ad-hoc algorithm.

The time signature $\binom{n_b}{b_\ell}$ is part of the input; we consider as basic time unit the length of the beats. We allow notes of the input to have a length greater than (actually, a multiple of) the beat length. However, if an input note is shorter than the beat length then we consider it only if the note is placed on the beat (otherwise it is only a passage note).

Let n_m denote the number of measures in the input bass line. We can represent a chromosome as an array C of dimension $4 \times n$, where $n \leq n_b \times n_m$. Element $C(i, j)$ contains the j^{th} note for voice i . We place the bass voice in the lower row and the soprano voice in the upper row. Notes are represented using MIDI codes; this allows us to easily play the chromosomes. We keep the information about notes' length in a separate data structure since we do not use such information for the genetic algorithm.

For each bass note b_i our goal is to select 3 notes, t_i, c_i, s_i , to be played along with the bass note b_i . The set of notes $\{b_i, t_i, c_i, s_i\}$ is the chord at position i . We can also think of a (candidate) solution to our problem as an array of $4 \times n$ cells, where each column i contains the four notes b_i, t_i, c_i, s_i . The set of these 4 notes must be the chord required by the figuration.

3.2 The Genetic Algorithm

Initial population. We start from an initial population that we build from the sample chromosome obtained from the ad-hoc algorithm applied to the input bass line (recall that we do not describe this algorithm in this paper). We get the initial population by choosing at random $K = 20$ chromosomes compatible with the sequence of chords provided by the ad-hoc algorithm. The sequence of chords has a “value” which is given by the sum of the weights in the graph representation (see previous section). This value is called the *base value* of the population.

Evaluation function. Our evaluation function takes into account musical composition rules from the common practice period. We consider a simple set of rules, which are detailed in Table 1.

To assign a value to a chromosome we start from the base value assigned to the sequence of chords; then we subtract a given cost for each rule violation. The relations between two consecutive genes (chords) allows us to decide whether the chromosome is good or not. For example, if a voice makes an augmented fourth jump we subtract 40 from the initial value; if two voices make a hidden fifth we subtract 60. Table 1 provides also the cost that we used for each rule. We construct the possible chords always putting voices in the order bass, tenor, alto and soprano; hence no voice crossing is allowed.

The fitness value of the chromosome is given by the base value of the population minus a cost which depends on the chromosome.

Table 1. Rules table

Single voice error	Cost	Type	Two voices error	Cost	Type
sixth	30	normal	hidden unison	40	normal
aug. fourth	40	normal	hidden fifth	60	normal
aug. fifth	40	normal	hidden octave	50	normal
seventh	80	critical	unison	100	critical
> octave	100	critical	parallel fifth	100	critical
			parallel octave	100	critical
Single voice cost	Cost	Type	Error within chord	Cost	Type
jump	interval	no error	octave leap	30	normal

Moreover, in the evaluation phase, our fitness function keeps some additional information regarding the pair of consecutive chords which contains more rule violations. These pairs are very important for the job of the evolution operators and we call these pairs *cut points*.

Evolution operators. In order to obtain the new population we apply both the crossover and the mutation. We consider all possible pairs of chromosomes in the current population and we apply the crossover operation to each pair. The crossover operation works as follow. Let C^1 and C^2 be the pair of chromosomes; for each cut point $C_i^1 C_{i+1}^1$ the crossover operator cuts in this point only if $C_i^2 C_{i+1}^2$ doesn't have critical rule violations and contains fewer normal rule violations than $C_i^1 C_{i+1}^1$, obtaining a new chromosome C^3 by letting $C^3 = C_0^1 \dots C_{i-1}^1 C_i^2 C_{i+1}^2 C_{i+2}^1 \dots C_n^1$. We remark that our cut points are not chosen at random, but are chosen to be points where the number of violations is higher. For each pair (C^1, C^2) we perform the crossover operation twice, swapping the roles of the two chromosomes.

The mutation operation takes each chromosome C in the initial population and produces a new chromosome D as follows. For each cut point $C_i C_{i+1}$ of C the mutation operator works in this point only if it can find a pair of randomly generated chords, $D_i D_{i+1}$, that contains no critical rule violation and fewer normal rule violations than $C_i C_{i+1}$, obtaining a new chromosome by letting $D = C_0 \dots C_{i-1} D_i D_{i+1} C_{i+2} \dots C_n$.

Selection method. At this point as a candidate new population we have the initial population of K chromosomes, at most $K(K - 1)/2$ new chromosomes obtained with the crossover operator and at most K chromosomes obtained with the mutation operator. Among these we choose the K chromosomes that have the best evaluation, according to the fitness function.

Stopping criteria. We stop the evolutionary process after a fixed number of generations (ranging from 100 to 500).

4 Implementation and Results

We have implemented the algorithm described in the previous sections using Java and the JGap library, for genetic algorithms subroutines, and the JFugue



Fig. 1. An example of input (part *a*), initial chromosome (part *b*) and final solution (part *c*)

library, for music manipulation subroutines. We have chosen as test cases inputs from bass lines taken from J.S.Bach's chorales.

We have run several tests repeating the test several times for the same input and varying the number of iterations of the genetic algorithm. Figure 1, part *a*, shows a bass line (input) taken from J.S.Bach's chorale BWV 70. The figuration found in the first stage is shown in part *b*; the positions of the voices in this chromosome are chosen at random. The output of the genetic algorithm is shown in part *c*.

In this particular example the evolution has corrected all critical errors. Actually in all tests that we have run all critical errors were corrected. In many tests, normal errors appeared also in the final solutions, but they were reduced.

5 Conclusions and Future Work

In this paper we have presented an automatic music composition algorithm based on a genetic algorithm. By automatic we mean that the composition is created without human intervention. The specific problem we considered is that of harmonizing a given bass line (figured or unfigured). Although this problem has already been considered in the literature most of the previous work uses different techniques. A few papers used genetic algorithms for problems similar to the one considered in this paper. One major difference with our work is that we use a specialized crossover and mutation operations. Moreover the works that use genetic algorithms and are closer to our work, do not consider the exact same problem considered in this paper.

The framework we have described in this paper (and the implementation of the algorithm) can be used to further study the use of genetic algorithms for the

specific problem of harmonizing a given bass line (and also for related problems). We plan to evaluate more sophisticated crossover and mutation operators and try to vary other important parameters (like the initial population for the genetic algorithm, the cost and the type of the errors considered, and other parameters that we used in the generation of the figuration) to evaluate their impact on the final solution. We have used simple (reduced) sets of rules and chords. Future work include the use of augmented sets of chords and rules.

Acknowledgments. We would like to thank Gianluca Zaccagnino for helping with the development of the Java implementation of the algorithm.

References

1. Biles, J.A.: GenJam: A genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference, pp. 131–137 (1994)
2. Cope, D.: Experiments in Musical Intelligence. A-R Editions (1996) ISBN 0895793377
3. Cope, D.: Virtual Music. MIT Press, Cambridge (2004)
4. Ebcioğlu, K.: An expert system for harmonizing four-part chorales. In: Machine models of music, pp. 385–401. MIT Press, Cambridge (1992)
5. Horner, A., Ayers, L.: Harmonization of musical progression with genetic algorithms. In: Proceedings of the International Computer Music Conference, pp. 483–484 (1995)
6. Horner, A., Goldberg, D.E.: Genetic algorithms and computer assisted music composition. Technical report, University of Illinois (1991)
7. Jacob, B.L.: Composing with genetic algorithms. Technical report, University of Michigan (1995)
8. Lehmann, D.: Harmonizing melodies in real time: the connectionist approach. In: Proceedings of the International Computer Music Conference, Thessaloniki (1997)
9. McIntyre, R.A.: Bach in a box: The evolution of four-part baroque harmony using a genetic algorithm. In: First IEEE Conference on Evolutionary Computation, pp. 852–857 (1994)
10. Phon-Amnuaisuk, S., Winnings, G.: The four-part harmonization problem: a comparison between genetic algorithms and rule-based system. In: Proceedings of the AISB 1999 Symposium on Musical Creativity, pp. 28–34 (1999)
11. Wiggins, G., Papadopoulos, G., Phon-Amnuaisuk, S., Tuson, A.: Evolutionary methods for musical composition. International Journal of Computing Anticipatory Systems (1999)
12. Schottstaedt, B.: Automatic species counterpoint. Tech. Rep. STAN-M-19, Stanford University CCRMA. In: Mathews, Pierce (eds.) A short report appeared in Current Directions in Computer Music Research. MIT Press, Cambridge (1989)

Generation of Pop-Rock Chord Sequences Using Genetic Algorithms and Variable Neighborhood Search

Leonardo Lozano, Andrés L. Medaglia, and Nubia Velasco

Departamento de Ingeniería Industrial
Centro de Optimización y Probabilidad Aplicada
Universidad de los Andes
`{leo-loza, amedagli, nvelasco}@uniandes.edu.co`

Abstract. This work proposes a utility function that measures: 1) the vertical relation between notes in a melody and chords in a sequence, and 2) the horizontal relation among chords. This utility function is embedded in a procedure that combines a Genetic Algorithm (GA) with a Variable Neighborhood Search (VNS) to automatically generate style-based chord sequences. The two-step algorithm is tested in ten popular songs, achieving accompaniments that match closely those of the original versions.

Keywords: Musical harmony, Genetic Algorithms, Variable Neighborhood Search.

1 Introduction

The problem of finding chord sequences for a given melody has been studied by some researchers in the last decade. For instance, Microsoft uses hidden Markov chains in the development of “MySong” [8], a software that generates chord sequences from vocal melodies. Other researchers have used combinatorial optimization [10] and Markov decision processes [1] as mathematical techniques to find “correct” chord progressions.

Nonetheless, determining when a chord progression can be labeled as “correct” is far from trivial and might be always open for discussion. From Chuan and Chew’s self-learning style based algorithm [1] to fitness functions for pleasant music based on Zipf’s law [5] or Schell’s “optimal criteria” based on traditional harmony rules [10], the challenge has been always to find a utility function capable of determining when a given solution is better than other.

This paper proposes a style-based utility function based on the pop-rock style that is comprised of *vertical* and *horizontal* utilities. This utility function is embedded in a procedure that automatically generates 3-note chord sequences.

2 The Pop-Rock Style Utility Function

The proposed utility function is made up of three main components. The first two measure the vertical relation among the chords in the sequence and the notes in the melody; while the third component measures the horizontal relation among the chords in the sequence.

2.1 Vertical Utility (Harmony)

To measure the vertical utility of a chord sequence (solution), the melody is divided into sections of equal length measured in bars. Each section is categorized and labeled according to its harmonic function into *Tonic*, *Dominant*, *Sub-Dominant*, *Not known* or *Non-tonality*. For this work, the allowed tonalities are all major tonalities.

A subjective scale was defined (Table 1) after a thorough analysis of multiple pop and rock songs, including those from classic bands like The Beatles, U2, and The Police, among others; and based on the musical harmony work by Kostka and Payne [4] and Piston and DeVoto [9]. This scale allows us to assign, for each interval between a note in a chord and a note in the melody, a utility value.

Table 1. Pop-rock utility values for all possible intervals

	Tonic, Subdominant and Not-Known sections	Dominant and Non-Tonality sections	Non-
Unison	4		4
Minor second	-3		-3
Major second	0		2
Minor third	3		3
Major third	3		3
Perfect fourth	2		2
Tritone	-5		3
Perfect fifth	2		2
Minor sixth	3		3
Major sixth	3		3
Minor seventh	0		2
Major seventh	-3		-3
Octave	4		4

Table 2. Pop-rock utility values for all possible chords in Tonic, Subdominant and Dominant Sections

Chord	Utility for Tonic sections	Utility for Subdominant sections	Utility for Dominant sections
I	5	-2	-3
ii	-2	4	-2
iii	2	-3	1
IV	-5	5	-2
V	-3	-2	5
vi	4	2	-2
vii°	-3	-2	1
Others	-3	-3	-3

A second subjective scale was defined (Table 2) to measure the coherence between each chord and the label given to the section. For example, if a section is labeled as “Tonic”, then a I chord will have more utility (5) than a V chord (-3).

In summary, the total vertical utility is computed using the vertical criteria for each note in the melody and each chord in the sequence.

2.2 Horizontal Utility (Chord Progression)

For evaluating the horizontal relation among chords, a list of twenty desirable chord progressions was built based on musical theory [3] and our own song analysis. Some of these progressions are the theoretically correct I-IV-V (i.e., Tonic, Sub-Dominant, Dominant) and the I-V-vi-VI progression, perhaps one of the most ubiquitous in pop-rock music. For each progression found in a given solution, the utility increases accordingly to its length. At the end, the total horizontal utility is given by the sum of the utility for all the progressions found in the solution.

3 The Proposed Two-Step Algorithm

The proposed two-step algorithm obtains a chord sequence considering both the vertical and horizontal utility dimensions. In the first step, a GA searches for a good set of solutions according to the vertical fitness function. In the second step, the best solutions obtained from the GA are used to build a network where a VNS algorithm finds a solution targeting the horizontal utility function. It is worth mentioning that even though there are two objectives, the algorithm uses a lexicographic approach where the vertical utility is considered more important than the horizontal utility. This is due to the fact that a dissonance between a note in the melody and a note in a chord is far more notorious than a poor chord progression.

3.1 The Genetic Algorithm Step

The input of the GA is comprised of three elements: a MIDI (Musical Instrument Digital Interface) file with the monophonic melody, the tonality of the melody, and the desired rhythm value for the chords. The genotype is a vector of chords that encodes the chord sequence. Each chord is represented by a pair of integers (gene), namely, the root of the chord and its quality. Figure 1 illustrates an example for a given chromosome. A fixed-size initial population is randomly created based on this solution encoding. Each parent from $P(t)$ is selected with probability p_c for crossover and each gene in the chromosome is uniformly mutated with probability p_m . Two types of crossover were tested: single-point and position-based crossover [7].

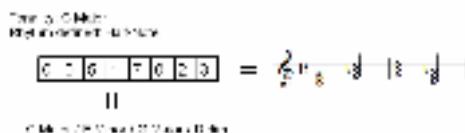


Fig. 1. Example of a four-gene chromosome

The goal of the GA is to feed with good vertical-utility accompaniments the second-step VNS algorithm, which in turn, focuses on improving the horizontal utility. To provide the VNS with a good assortment of choices, M independent replications (runs) of the GA are conducted. The best solutions for these replications provide different options for the same section of the melody. Based on these solutions, an output network is created with all the possible solutions for each section as shown in Figure 2. In this network, each node corresponds to a chord that appears in at least one solution obtained from the GA.

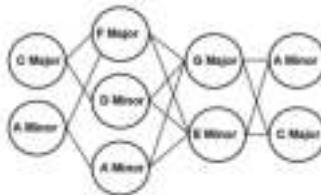


Fig. 2. Example output network of the first-step GA (input network for the second-step VNS)

3.2 The Variable Neighborhood Search Step

VNS [2] is used to improve upon the solutions found by the GA in terms of the horizontal utility. The objective is to find a path on the network generated by the GA (see Figure 2) that contains the maximum number of large pop-rock progressions, defined by the horizontal utility function. Starting from a random path, the neighborhood structure is defined by fixing nodes in some sections of the path, while freely exchanging nodes in the remaining sections. The local search algorithm of the VNS finds and matches nodes that belong to a progression that increases the horizontal utility of the solution. These nodes have a high probability p_f of remaining fixed (correspondingly only $1 - p_f$ chance of being changed); while the rest of the nodes are randomly exchanged with those in their corresponding section.

4 Computational Results

The algorithm was implemented on JGA [6], a Java-based tool for rapid development of GAs and using the jMusic framework, a class (object) library written for musicians in the Java programming language [11].

This section illustrates the two-step algorithm described in §3 (henceforth called GA+VNS), by finding evolutionary chord sequences to popular melodies. For each melody, the original chord sequence was used as its own benchmark. The algorithm performance was measured based on the fraction of chords that exactly match the original (called *Exact Match*); and the fraction of chords that exactly match the original or are a Major/Minor relative chord (called *Relative Match*). All the experiments were conducted on an Intel Pentium IV CPU running at 3.0GHz with 1024 MB of RAM under Windows XP Professional.

Table 3 shows the melodies' titles, authors, and their length; the population size (P), the maximum number of iterations (N), and the crossover type; the exact and

Table 3. Results for ten test melodies

Song		Length (Notes)	P	N	Crossover r	Exact Match	Relative Match	Time (s)
Twinkle Twinkle Star (Jane Taylor)	Little	13	100	100	Position-Based	87%	100%	30
Let It Be (The Beatles)		25	100	100	Position-Based	63%	75%	64
I'm Like a Bird (Nelly Furtado)		28	20	20	Position-Based	25%	50%	28
Mariposa Technicolor (Fito Paez)		28	100	100	Position-Based	60%	85%	72
Every Breath You Take (The Police)		30	40	30	Position-Based	71%	71%	9
Basket Case (Green Day)		32	30	30	Single-Point	50%	67%	10
Tears in Heaven (Eric Clapton)		48	100	100	Position-Based	41%	50%	92
Yellow (ColdPlay)		58	40	30	Position-Based	54%	62%	30
Buscando un Poco de Amor (Shakira)		102	60	60	Single-Point	65%	65%	92
I Don't Wanna Miss a Thing (Aerosmith)		132	50	50	Position-Based	50%	53%	250

relative match and the execution time. The crossover (p_c) and mutation (p_m) rates were set at 0.9 and 0.2 for all instances. The input melodies and the generated accompaniments by GA+VNS are available at <http://hdl.handle.net/1992/1092>.

Figure 3 compares the original solution and the generated solution by GA+VNS on Mariposa Technicolor by Fito Paez. For this song, the author uses two chords outside the tonality. In the case of the A Major chord, there is not a leading tone (C sharp) in the melody so the generated solution uses a G Major (guided by the G and B notes in the melody). In the latter case of the B Major chord, the lead tone D sharp is present in the melody and the generated solution includes the B Major chord and its correct resolution to E Minor.

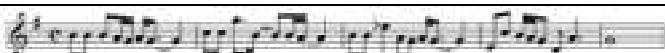
Original	G D Em Em C G A D G B Em Em C C D D G G G G
GA+VNS	G G Em G Am G G D G B Em G C Em D D G Em G
Melody	

Fig. 3. Original vs. GA+VNS chord sequence for “Mariposa Technicolor”

In most cases, the solutions generated by GA+VNS show high exact and relative matches, resulting in good chord progressions. However, in few cases when the input melody is repetitive (e.g., The Beatles and The Police), the resulting chord progression is static and boring, despite the fact that it presents high relative match. In other cases (e.g., Furtado and Aerosmith) it happens just the opposite, that is, the

exact and relative matches are low, yet the resulting chord progression is pleasant and blends well with the melody.

5 Conclusions and Future Research

Finding the right chord sequence for a melody requires years of musical study and, even for professional musicians, it is far from trivial. The proposed two-step algorithm generates pop-rock style-based accompaniments for any given melody. First, it uses a GA that searches for solutions with a high vertical utility. Second, with the best solutions of the GA, a network is created and explored with VNS to improve the horizontal utility. The algorithm was tested in ten instances of pop rock music. In most cases the resulting accompaniments highly match those of the original songs.

The proposed method is especially useful for amateur musicians who may want to generate chord sequences for their melodies in a reasonable time using evolutionary powered software.

In the future, the algorithm will be extended by defining theoretically-supported utility functions for different genres like blues and jazz; including seventh and ninth chords; and finding the proper chords on more difficult static melodies.

References

- Chuan, C., Chew, E.: A Hybrid System for Automatic Generation of Style-Specific Accompaniment. In: Proceedings of the Fourth International Joint Workshop on Computational Creativity, University of London (2007)
- Hansen, P., Mladenovic, N.: Variable Neighborhood Search: Principles and Applications. European Journal of Operational Research 130, 449–467 (1999)
- Harrison, M.: Contemporary Music Theory Level Two. Hal Leonard, Milwaukee (2004)
- Kotska, S., Payne, D.: Tonal Harmony with an Introduction to Twentieth-Century Music. McGraw-Hill, New York (2003)
- Manaris, B., Machado, P., McCauley, C., Romero, J., Krehbiel, D.: Developing Fitness Functions for Pleasant Music: Zipf's Law and Interactive Evolution Systems. In: EvoMUSART 2005, 3rd European Workshop on Evolutionary Music and Art, Lausanne, Switzerland (2005)
- Medaglia, A.L., Gutiérrez, E.: JGA: An Object-Oriented Framework for Rapid Development of Genetic Algorithms. In: Rennard, J.-P. (ed.) Handbook of Research on Nature Inspired Computing for Economics and Management. Grenoble (2006)
- Michalewicz, Z.: Genetic Algorithms Plus Data Structures Equals Evolution Programs. Springer, New York (1996)
- Morris, D., Basu, S., Simon, I.: MySong: Automatic Accompaniment Generation for Vocal Melodies (2008)
- Piston, W., DeVoto, M.: Harmony. Editorial Labor, Barcelona (1992)
- Schell, D.: Optimality in Musical Melodies and Harmonic Progressions: The Traveling Musician. European Journal of Operational Research 140(2), 354–372 (2002)
- Sorensen, A., Brown, A.: jMusic: Musical Composition with Java (website). (last accessed on: November 9, 2008), <http://jmusic.ci.qut.edu.au/index.html>

Elevated Pitch: Automated Grammatical Evolution of Short Compositions

John Reddin¹, James McDermott², and Michael O'Neill²

¹ Trinity College Dublin

reddinj@tcd.ie

² University College Dublin

jamesmichaelmcdermott@gmail.com, m.oneill@ucd.ie

Abstract. A system for automatic composition using grammatical evolution is presented. Music is created under the constraints of a generative grammar, and under the bias of an automatic fitness function and evolutionary selection. This combination of two methods is seen to be powerful and flexible. Human evaluation of automatically-evolved pieces shows that a more sophisticated grammar in combination with a naive fitness function gives better results than the reverse.

Keywords: Grammatical evolution, music, automatic composition.

1 Introduction

Automatic musical composition is a topic of both theoretical and practical interest, and evolutionary computation (EC) has been used for this task with some success. One approach which has received relatively little attention is EC guided by generative grammars — for example, grammatical evolution or GE [1]. There is a strong motivation for this approach, since generative grammars have been used extensively in music theory and analysis.

In this paper, we present a system for automatic grammatical composition and consider some key questions in this area. We introduce GE with an example of a grammar for generating music, and look at methods of implementing automatic fitness functions. This paper thus responds to McCormack's call [2] for research into automatic fitness functions. Since grammars can be open-ended and recursive, this paper also responds to Bentley's call [3] for open representations in creative domains. The chief contributions are a demonstration that GE is a method suitable for automatic composition, and a set of experiments with human evaluation of automatically-composed pieces.

2 Previous Work

There is a significant body of work in the area of generative composition. EC is a popular paradigm, whether using automatically-calculated fitness (e.g. [4]) or interactively (e.g. [5]). The ideas are not limited to academia [6].

However, a great deal of work remains to be done. Systems which use unstructured representations may be capable of producing all desired pieces, but at the cost of making them “needles in a haystack”. A naive implementation of a linear-genome GA for composition might have each integer gene giving the pitch value of a directly-corresponding note. Such a GA could represent all possible melodies (for a fixed note duration), but the vast majority of these melodies will feel (at best) somewhat meandering.

An alternative is to impose structure on the search space using a formal grammar. Grammars are well-established as tools of musical analysis and creation [7,8,9]. There is evidence that listeners perceive in music the hierarchical structures characteristic of generative grammars [9]. The use of grammars addresses a key problem suffered by simple linear representations of music, which is that the compositional form is fixed. Representations for creative evolution may benefit from being open-ended [3]. Mapping via a grammar can lead to a piece of music of any number of voices, or any duration — as decided by evolution.

To our knowledge, Ortega and co-authors [10] are the only researchers to report work using GE to compose music. This paper was a useful proof-of-concept, though the key issues such as grammar design, fitness functions, and subjective evaluation were left unaddressed.

3 Introduction to Grammatical Evolution

GE is a type of genetic programming which uses generative grammars to define the possible forms of “programs” [1]. A grammar consists of multiple rules, each with a left-hand-side (LHS) and multiple possible productions. Derivation begins with a specified LHS. At each step, the first LHS in the derived string is replaced by one of its corresponding productions. The choices of productions are made according to successive values in the linear integer-valued genome. Two simplified fragments from the grammars described in Sect. 4.2 are given in Figs. 1 and 2.

```
<melody> ::= <chord_material><bass_material><melody_material>
<melody_material> ::= <bar><bar><bar><bar>
<bar> ::= <note_or_rest><note_or_rest><note_or_rest><note_or_rest>
<note_or_rest> ::= <quaver> | <quaver_rest> | <semiquaver><semiquaver>
<quaver> ::= <midi_pitch> 0.5
```

Fig. 1. A fragment of a simple musical grammar. 0.5 indicates the quaver’s duration.

```
<melody_material> ::= <bar><bar_or_op><bar_or_op><bar_or_op>
<bar_or_op> ::= <bar> | <op>
<op> ::= COPYPREVBAR | SHUFFLEPREVBAR | TRANSPOSEPREVBAR <n>
      | SHUFFLEPREVBAR | REVERSEPREVBAR | INVERTPREVBAR
      | COPYPREVBARTRANSPOSEFINALNOTE <n>
```

Fig. 2. A fragment of a grammar with transformations. <n> indicates an integer-valued transposition offset.

4 Experimental Setup

The overall aim of the experiments was to compare human judgement of pieces produced by contrasting GE setups. Four experiments are to be described: each consisted of two stages, as follows.

Firstly, in each case, two sets of ten short pieces were produced according to two contrasting automatic evolution schemes. These schemes differed in each case by the grammar, the fitness function, or the contrast between evolution and mere random generation — as explained in Sect. 5.

A typical GE setup was used, with a population size of 200 and 300 generations. Fixed-point crossover had a probability of 0.9 and the mutation probability was 0.01. Selection was roulette-wheel, and replacement was generational with 1-individual elitism. The chromosome size was 500 and two occurrences of the GE wrapping operator were allowed per individual.

Secondly, in each case, the ten pairs of pieces were presented to multiple subjects for listening and evaluation. The order within each pair was randomised. The number of volunteers for the four experiments was between 13 and 22. Subjects were required to state a preference for one track or the other.

4.1 Fitness Functions for Musical Composition

The question of automated fitness functions is a central one in evolutionary art and music [2], and one which will likely never be fully resolved. Perhaps the best that can be hoped for is to produce a fitness function which penalises obviously undesirable qualities on a limited domain: as Dahlstedt [4] says, automatic fitness functions can be used to weed out bad results, relying on the expressive power of the representation to produce good ones.

With this in mind, we immediately define one simple fitness function for short pieces of music by fixing the desired scale (say, C major) in advance, and counting the notes in the piece which are not in this scale. By minimising this number through evolution, it is possible to achieve pieces which conform to the simple requirement that the piece be strictly in the desired key. However, this fitness function is clearly insufficient in itself. There are many possibilities for more sophisticated measurement of the overall “quality” of a piece of music.

Towsey and co-authors defined [11] a set of 21 melodic measures, each of which calculates the value of a statistic over a melody. For example, *pitch range* subtracts the lowest from the highest pitch, and *note density* calculates the proportion of notes to rests. These measures were calculated over a corpus of “good” melodies to give an allowable range for each measure.

This approach is adopted here for our second fitness function. We chose the 6 measures given by Towsey et al. with the lowest standard deviations across the set of melodies they used, omitting two inapplicable ones (*rhythmic variety* and *rhythmic range*), and replacing them with two more measures with very low standard deviations. The 6 measures to be used are summarised in Table 1.

Fitness was formulated based on these measures in a method similar to that of Dahlstedt [4]. Given a desired value v_d , and a standard deviation s , values

Table 1. Melodic measures

Measure	Desired value	Standard deviation
Pitch variety	0.27	0.11
Dissonance	0.01	0.02
Pitch contour	0.49	0.06
Melodic direction stability	0.40	0.11
Note density	0.49	0.15
Pitch range	0.50	0.10

$v \in (v_d - s, v_d + s)$ within the desired range were awarded a component-fitness value of 0. Values below the range were awarded a component-fitness value of $\sqrt{(v_d - s) - v}$, whereas values above the range were awarded a component-fitness value of $\sqrt{v - (v_d + s)}$. The component-fitness values were then averaged across all six measures (giving a result in $[0, 1]$), and the result was averaged with that returned by the simple fitness function described above.

4.2 Musical Grammars

Two types of grammars were used in experiments. An *unstructured* grammar (as in Fig. 1) simply states that a piece consists of N voices, a voice consists of M bars, a bar consists of 4 beats, and each beat consists of either a quarter-note or two eighth-notes, and each note consists of a note proper, or a rest. Such a grammar imposes very little structure on the search space — in fact, it is equivalent in a sense to the linear-genome GA mentioned earlier.

A *structured* grammar (as in Fig. 2) was designed which takes advantage of some aspects of musical knowledge available to all composers. For example, when composing a melody, it is common to repeat a sequence (perhaps with alteration), to shuffle previous notes, to invert or reverse a sequence or to transpose a sequence. These transformations were implemented in the grammar so that when the mapping process chose to express a transformation, it operated on the material of the previous bar and the result lasted exactly one bar. This is a simple implementation to be expanded in future work.

5 Results

Some example tracks, together with the experimental software, are available for download from http://www.skynet.ie/~jmmcd/elevated_pitch.html

Experiment 1 compared random generation and evolution. A 16-bar unstructured grammar was used, with a simple fitness function. Fig. 3(a) shows the results. For each user, the number of preferences of unevolved and evolved tracks were summed: the boxplots represent the distribution of these sums. The evolved tracks were preferred much more often.

Experiment 2 compared (8-bar) unstructured and structured grammars using the simple fitness function. Fig. 3(b) shows the results. There was a preference for

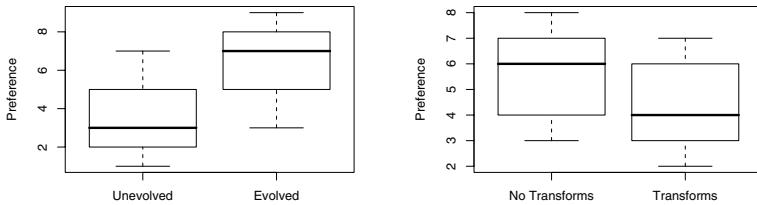


Fig. 3. Results from experiments 1 and 2. The evolved pieces were preferred to the unevolved ones, but the addition of transformations to the grammar was not successful.

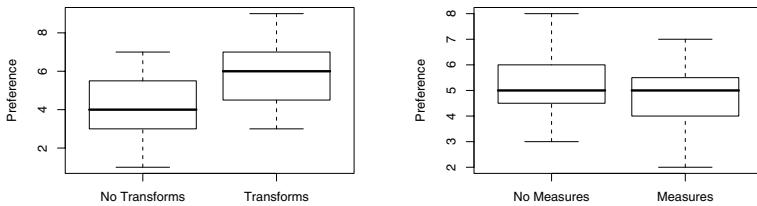


Fig. 4. Results from experiments 3 and 4. This time, transformations made a big improvement, but additions to the fitness function made things worse.

the outputs of the unstructured grammar — a surprising result. The hypothesis was formed that the short length of the pieces in this experiment, together with the possibility that the melody would rest in the first 2 or 4 bars, was hiding the shortcomings of the unstructured grammar. This led to the design of new grammars similar to those used in the previous experiment, but producing 12-bar pieces and preventing the melody from remaining silent in the initial bars.

Experiment 3 compared these 12-bar unstructured and structured grammars, again with the simpler fitness function. The results are shown in Fig. 4(a). Users preferred the structured grammar, with results significant at the $p < 0.05$ level.

Finally, experiment 4 compared the simple fitness function and the more sophisticated one using melodic measures. A 12-bar unstructured grammar was used. The results (in Fig. 4(b)) turned out to be worse for the fitness function using melodic measures, contrary to expectation. One possible explanation is that here automatic evolution, in attempting to satisfy multiple objectives, failed to eliminate all non-scale notes from the pieces.

6 Conclusions

Through human evaluation of automatically-composed pieces, GE has been shown to be capable of producing results which are better than random generation, and it has been shown to be sufficiently flexible to be extended in multiple directions, through constraints (in the grammar) and biases (in the fitness function), separately or together. This flexibility is a great advantage of GE.

The strongest results appeared in the third experiment: subjects were able to recognise that structure had been imposed on melodies and several subjects reported that they enjoyed this experiment more than others. One subject commented “[...] after the first few plays, I feel like I began to detect how the transformations were occurring.” This is a positive result.

Our work in using GE in creative domains is ongoing. Among many possibilities we mention four for potential future work: better automated fitness functions; the comparison of the grammatical approach with state-of-the-art implementations of other methods, such as tree-genome GP and N-gram models; more complex grammatical models; and interactive GE for musical composition.

Acknowledgments. John Reddin is funded under Science Foundation Ireland's UREKA scheme. James McDermott receives Irish Research Council for Science, Engineering and Technology post-doctoral funding.

References

1. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, Dordrecht (2003)
2. McCormack, J.: Open problems in evolutionary music and art. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2005*. LNCS, vol. 3449, pp. 428–436. Springer, Heidelberg (2005)
3. Bentley, P.J.: Exploring component-based representations - the secret of creativity by evolution? In: Parmee, I.C. (ed.) *Proceedings of ACM 2000*, University of Plymouth, pp. 161–172 (2000)
4. Dahlstedt, P.: Autonomous evolution of complete piano pieces and performances. In: *Proceedings of Music AL Workshop* (2007)
5. Nelson, G.L.: Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms (1993), <http://timara.con.oberlin.edu/~gnelson/gnelson.htm>
6. Eno, B.: A Year with Swollen Appendices. Faber and Faber (1996)
7. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press, Cambridge (1983)
8. Kippen, J., Bel, B.: Modelling music with grammars: Formal language representation in the BOL processor. In: Pople, A.M.A. (ed.) *Computer Representations and Models in Music*, pp. 207–238. Academic Press, London (1992)
9. Marsden, A.: Generative structural representation of tonal music. *Journal of New Music Research* 34(4), 409–428 (2005)
10. de la Puente, A.O., Alfonso, R.S., Moreno, M.A.: Automatic composition of music by means of grammatical evolution. In: *Proceedings of APL 2002*, Madrid (2002)
11. Towsey, M., Brown, A., Wright, S., Diederich, J.: Towards melodic extension using genetic algorithms. *Educational Technology & Society* 4(2) (2001)

A GA-Based Control Strategy to Create Music with a Chaotic System

Costantino Rizzuti¹, Eleonora Bilotta¹, and Pietro Pantano²

¹ Department of Linguistics, University of Calabria, Cubo 17b Via P. Bucci,
Arcavacata di Rende (CS) 87036, Italy
bilotta@unical.it, costantino.rizzuti@unical.it

² Department of Mathematics, University of Calabria, Cubo 30b Via P. Bucci,
Arcavacata di Rende (CS) 87036, Italy
piepa@unical.it

Abstract. Chaotic systems can be used to generate sounds and music. Establishing a musical interaction with such systems is often a difficult task. Our research aims at improve the extent of interaction provided by a generative music system by using an evolutionary methods. A musician can hear and imitate what the generative system produces; therefore, we are interested in defining a control strategy to allow the generative music system to imitate the musical gesture provided by the musician.

1 Introduction

Since the 80s many musical researchers have tried to use chaotic non-linear dynamical systems as melodic pattern generators. Many different characteristics make Generative Music using chaotic systems an attractive field of research, both for musicians and scientists [1,2].

There is a growing body of work involving the use of Genetic Algorithms (GAs) in musical applications [3,4]. GAs have been also used to create interactive systems [3] able to realize a real-time interactive performance.

In our previous work [5], a Genetic Algorithm has been used to explore the space of parameters of the Chua's oscillator to select melodic sequences. In this paper we present a GA-based evolutionary method to control a generative music system in order to obtain melodies with given properties.

The paper is organized as follows: Section 2 presents the interactive generative music system. Section 3 exposes the Genetic Algorithm implementation. Section 4 presents the method we used in testing the system, and provides some results. In Section 5 we conclude the paper showing the future directions of this work.

2 Interactive Generative Music

In a broad sense “Generative music” refers to music that is created in algorithmic manner, however this term is often used with different meanings. In this work we will present an algorithmic music system based on chaotic systems to create

musical materials. According to Wooller [6], we will use the term “Generative music” mainly referring to systems that output data with an increased size and a more general musical organization in comparison to the input. A system that creates music using complex dynamical systems can be mainly divided in three parts. A dynamical system generating a time series, a codification system allowing the time series to be translate into musical parameters, and a synthesis engine transforming these parameters into music.

2.1 Chua’s Oscillator

Chua’s oscillator [7] is a canonical system for research in chaos, since it can be realized in a real-world setting as a simple electronic circuit. The system has three degrees of freedom, and is described by three state equations. However, we use a dimensionless model to simulate it, with six parameters: $\alpha, \beta, \gamma, a, b, k$ corresponding to the ratios between two or more physical components.

The state equations for the dimensionless model are:

$$\begin{cases} \frac{dx}{d\tau} = k\alpha[y - x - f(x)] \\ \frac{dy}{d\tau} = k(x - y + z) \\ \frac{dz}{d\tau} = k(-\beta y - \gamma z) \end{cases}$$

where:

$$f(x) = bx + \frac{1}{2}(a - b)\{|x + 1| - |x - 1|\}$$

Solving this system of integral equations allows us to simulate the evolution of the Chua’s oscillator in three-dimensional phase space, made up by the variables: x, y , and z . The qualitative behavior of the oscillation is controlled by the six input parameters previously mentioned.

2.2 Codification

Many mapping strategies can be used to create an auditory representation of the time series produced by a dynamical system [8,9]. The method used in this work allows us to map the evolution of a time series into a sequence of values of a musical parameter (i.e. pitch, duration). By using this method it is possible to define the vocabulary of values (the set of admissible elements) to be used in creating the sequence. For the generation of musical pitches, for example, it is important to define the ordered set S_h containing the h possible notes that can be generated:

$$S_h = \{note_1, note_2, \dots, note_h\}.$$

The codification system perform a quantization of the time series generated by the chaotic system to fit with the number of possible notes. The notes are synchronously generated by sampling the chaotic time series using a certain sampling period.

3 Evolutionary Process

A genetic algorithm has been used to explore the space of parameters of the Chua's oscillator to find the set of parameters that allows it to generate the melodic sequence that is more similar to a given target melody. Fig. 1 shows how the genetic algorithm acts as a control for the generative music system.

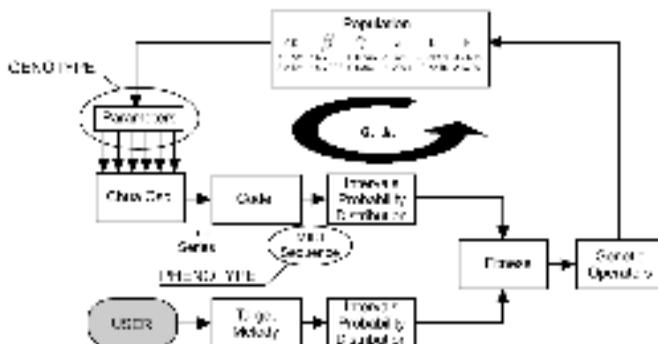


Fig. 1. The Genetic Algorithm architecture

The genetic algorithm evolves a population of artificial genotypes corresponding to the six parameters of the Chua's oscillator. Then, the GA search looks for a combination of parameters of the Chua's oscillator that define a phenotype (a sequence of notes) matching a target melody according to a certain metric.

3.1 Fitness Function

The fitness function we have used corresponds to a measurement of the similarity between two different sequences of notes. We are not interested in finding a perfect imitation of the target melody. For this reason, we have chosen to use an high level index based on statistical measurement, that allows us to measure the similarity of the musical sequences without tightly consider the actual sequence of notes in the target and in the evolved melodies. We have defined an evaluation function based on a comparison between the probability distribution of the musical intervals in the target melody and those related to the melodies generated by using the generative music system.

In order to evaluate the fitness of each individual in the population we firstly generate a melodic sequence of 200 notes and we evaluate the probability distribution of the musical intervals occurring in this melody, then we compare the obtained probability distribution with the distribution computed for the target melody. The comparison between the two probability distributions can be made by using the Kullback–Leibler divergence (KL)[11], that is a non-commutative measure of the difference between two probability distributions. For probability

distributions P and Q of a discrete random variable the KL divergence of Q from P is defined to be:

$$D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

The Kullback–Leibler divergence has the property to be always non-negative and it is also equal to zero if and only if $P = Q$, so we use KL divergence as the evaluation function to measure the fitness of each individual. The fitness of an individual will increase if the value computed by the evaluation function will approximate values near to zero.

3.2 Genetic Algorithm Implementation

The population has been defined as an array containing the genotypes, made up of six real numbers for the parameters of the Chua's oscillator, and an extra real number to store the result of the evaluation function. The population size has been fixed at 25 genotypes (or individuals). In each cycle of the GA a new generation is computed by using a suitable breeding strategy. For the generation of the offspring we have used the “roulette wheel selection strategy” to choose the individuals to be subjected to crossover. Crossover operation is done using a pool made up of the fittest individuals of the current population. The size of the pool has been fixed to be equal to half the size of the population. The crossover operator has been implemented as one point crossover between two randomly chosen genotypes. The mutation operator is randomly applied with a certain probability to a new individual created by crossover. The mutation is realized as a random variation affecting only one gene randomly chosen.

4 Test and Results

We used four different intervals probability distributions to test the performance of the Genetic Algorithm. The first target is a sequence of an ascending and a descending tone (i.e. C-D-C-D-...). The second target is similar to the first but the repetition of the note with the same probability of the two whole tone intervals has been added. The third target is a sequence of an ascending and descending major third (i.e. C-E-C-E-...). The forth target is similar to the second one, adding the major third intervals.

Using the MIDI standard all the possible intervals within two octave can be represented by the numbers defined in the range $[-12, 12]$. The graphics in Fig. 2 show the probability distributions obtained by five different runs of the GA evolved for 200 iterations. In particular, the graphic on the left refers to the first target; that on the right is related to the second target. The graphics in Fig. 3 show the probability distributions obtained by five different runs of the GA evolved for 1000 iterations. In particular, the graphic on the left refers to the third target; that on the right is related to the forth target. The results show a good match between the targets and the melodies generated through the

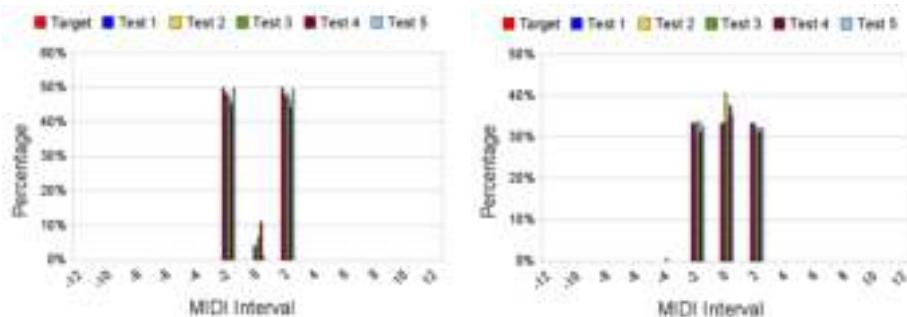


Fig. 2. Intervals probability distributions for the first target (left) and for the second (right)

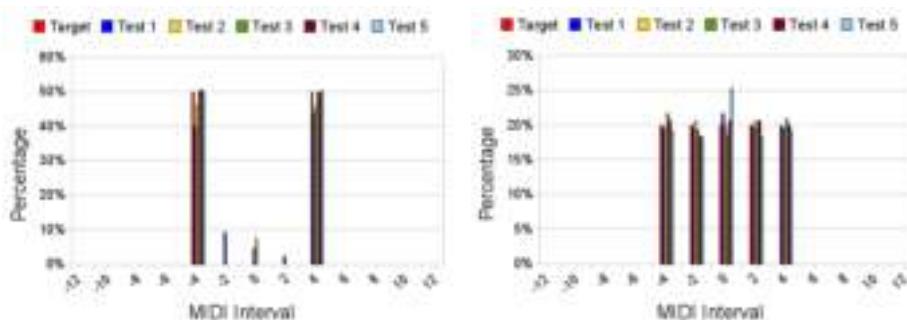


Fig. 3. Intervals probability distributions for the third target (left) and for the forth (right)

GA. However, the graphics in Fig. 2 and in Fig. 3 show also that many evolved melodies are characterized by intervals which don't belong to the target. These extra intervals have often a very small probability to occur. From a musical point of view, this result can be very interesting, because the selected melodies are imitations with some degree of variation of the target melody.

5 Conclusion

In this paper we have presented an evolutionary method to control a generative music system based on a chaotic system. A genetic algorithm was used to control the parameters of the Chua's oscillator in order to recreate a target melody.

The obtained results are very encouraging because they show that it is possible to control the parameters of the Chua's oscillator in order to produce a musical sequence with certain properties. However, only simple targets have been tested and the search space has been limited to a narrow portion on the Chua's oscillator parameters space. Moreover we need to evaluate the results from a musical point of view and to explore the interaction opportunity provided by such system.

In future work we will test the performance of the GA with musically more complex targets. We will also try to use the genetic algorithm within a interactive generative music system to verify if the GA-based control improves the extent of interaction and the kind of feedback the system is able to provide to a musician. We are lanning to test different scenarios: running the genetic algorithm offline before a performance, run the algorithm at certain moments during a performance, or running it repeatedly in realtime.

References

1. Pressing, J.: Nonlinear Maps as Generators of Musical Design. *Computer Music Journal* 12(2), 35–45 (1988)
2. Bidlack, R.: Chaotic systems as Simple (but Complex) Compositional Algorithms. *Computer Music Journal* 16(3), 33–47 (1992)
3. Miranda, E.R., Biles, J.A. (eds.): *Evolutionary Computer Music*. Springer, London (2007)
4. Gartland-Jones, A.: MusicBlox: A Real-Time Algorithmic Composition System Incorporating a Distributed Interactive Genetic Algorithm. In: Cagnoni, S., et al. (eds.) *EvoWorkshops 2003*. LNCS, vol. 2611, pp. 490–501. Springer, Heidelberg (2003)
5. Bilotta, E., Pantano, P., Cupellini, E., Rizzuti, C.: Evolutionary Methods for Melodic Sequences Generation from Non-linear Dynamic Systems. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 585–592. Springer, Heidelberg (2007)
6. Wooller, R., Brown, A., Diederich, J., Miranda, E.R., Berry, R.: A framework for comparison of process in algorithmic music systems. In: *Proceedings of the Generative Arts Practice - A Creativity and Cognitinion Symposium*, Sydney (2005)
7. Chua, L.O.: Global Unfolding of Chua Circuits. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E76-A, 704–734 (1993)
8. Rizzuti, C.: Mapping Chaotic Dynamical Systems Into Timbre Evolution. In: *Proc. of Sound and Music Computing Conference (SMC 2007)*, Lefkada, Greece, National and Kapodistrian University of Athens, pp. 22–29 (2007)
9. Bilotta, E., Cooperstock, J.R., Cupellini, E., Pantano, P., Rizzuti, C., Wozniewski, M.: Exploring Musical Mappings and Generating Accompaniment with Chaotic Systems. In: *Proceedings of International Computer Music Conference (ICMC)*, Belfast, pp. 467–474 (2008)
10. Winkler, T.: *Composing Interactive Music - Techniques and Ideas Using Max*. MIT Press, Cambridge (2001)
11. Kullback, S., Leibler, R.: On information and sufficiency. *Annals of Mathematical Statistics* 22, 79–86 (1951)

Teaching Evolutionary Design Systems by Extending “Context Free”

Rob Saunders and Kazjon Grace

Sydney University, Sydney NSW 2006, Australia

rob@arch.usyd.edu.au, kazjon.grace@usyd.edu.au

<http://web.arch.usyd.edu.au/~rob>

Abstract. This document reports on a case study using a novel approach to teaching generative design systems. The approach extends Context Free, a popular design grammar for producing 2D imagery, to support parametric and evolutionary design. We present some of the challenges that design students have typically faced when learning about generative systems. We describe our solution to providing students with a progressive learning experience from design grammars, through parametric design, to evolutionary design. We conclude with a discussion of the benefits of our approach and some directions for future developments.

1 Introduction

An understanding of the principles of *generative design systems* is an important component of any modern education in computer-aided design. The application of generative design technologies to solve complex real-world design problems has come to public attention with the development of the “bird’s nest” stadium for the Beijing Olympics [1]. Generative design systems are also increasingly finding uses in the “mass customisation” of goods and services, where they can be used to produce custom designs for anything from clothing to jewellery to housing. Technologies used to create generative design systems including a range of computational processes including evolutionary systems.

1.1 Teaching Generative Design Systems

As part of the Bachelor in Design Computing at the University of Sydney, we have been teaching generative design systems for more than five years, to both undergraduate and graduate students. These students typically have a strong design focus and, while their courses include programming, usually do not possess the breadth or depth of a computer science student. The curriculum for Generative Design Systems introduces concepts including; Expert Systems, Design Grammars, Parametric Design, and Evolutionary Design.

In the past we have had students implement their own evolutionary design tools based on a skeleton implementation. Based on student feedback, it was evident that the necessity to implement custom design domains significantly limited

student engagement. Students cited confusion caused by the number of “new” concepts introduced by evolutionary design systems. Many students were able to conceptualise and design the problems and fitnesses they wished to implement but lacked the technical ability to implement them. We also observed that using dedicated tools to teach individual generative design concepts created a barrier to students integrating concepts towards a more complete understanding.

To address these issues we have used and extended the Context Free design tool to provide a single environment to learn about three important concepts; design grammars, parametric design, and evolutionary design. The remainder of this paper presents how we have used this system provide a path for learning about design grammars, parametric design and evolutionary design through a series of simple extensions of the initial grammar.

2 Description

The Context Free Design Grammar (CFDG) is a simple language designed to generate complex images from simple programs [2]. Context Free is an open-source graphical development environment for writing and rendering CFDG programs [3]. Informally, a CFDG program contains a set of simple rules describing how to draw designs using a small number of pre-defined terminal shape rules including `circle`, `square`, and `triangle`. Recursive grammars include rules that call themselves. Non-deterministic grammars include multiple rules with the same name, the rule to execute is chosen probabilistically. An example of a recursive, non-deterministic CFDG program together with one of the images it can produce is given in Fig. 1.

Other graphical grammar-based design tools such as LParser [4] or Xfrog[5] have more powerful rendering engines and can produce 3D models as well as 2D images. The advantage of Context Free is its simple syntax and intuitive user interface. Our experience has been that students are able to explore the generative potential of recursive and non-deterministic rules and rapidly develop proficiency by modifying, extending and creating Context Free grammars despite their limited programming experience.

2.1 Parametric Context Free

By definition, CFDG does not support variables or parameters; the inclusion of variables would violate the need to keep the CFDG rules free of context. To allow students to explore parametric design grammars we first created an interpreter for parametric design grammars that uses template files to define a set of parameters and a set of modified CFDG rules.

An example template is shown in Fig. 2a. This template defines ten parameters and three rules: the rule `LINKS1` calls `LINKS2` and then recursively calls itself, similarly `LINKS2` calls `LINK` and then recursively calls itself, `LINK` draws a short line segment using the primitive `TRIANGLE` rule that draws an equilateral triangle. Parameters are defined within the comments tags at the top of the file

```

startshape SEED1

rule SEED1 {
    SQUARE{}
    SEED1 {y 1.0 s 0.99 r 1.5}
}

rule SEED1 0.05 {
    SQUARE{}
    SEED1 {y 1.0 s 0.99 r 1.5}
    SEED1 {y 1.0 s 0.6 r -60}
    SEED1 {y 1.0 s 0.5 r 60}
}

rule SEED1 0.05 {
    SEED1 { flip 90 }
}

```

(a) Source code



(b) Generated image

Fig. 1. An example Context Free Design Grammar and one of the images that it can generate. Context Free uses short letter strings to represent its random number seeds. This image was generated using the string MKY.

and referenced throughout the rules to define scales, rotations, and translations. Standard CFDG grammar files are generated by substituting all references to a parameter with a value within the range defined for the parameter. Some of the possible outputs for the parametric CFDG in Fig. 2a are shown in Fig. 2b. The space of possible designs generated from the three CFDG rules (comprising just seven lines of code) is very large, illustrating how quickly students could generate interesting design spaces to explore with the parametric CFDG system.

2.2 Evolutionary Context Free

To allow students to evolve design grammars, an evolutionary CFDG system was developed using the parametric CFDG system. In the evolutionary CFDG system, a parameter set in a template file defines the genotype for individuals and the rules are used to express genotypes into phenotypes. An interactive evolutionary system was first developed, where the phenotype (image) for each individual in a population is displayed side-by-side. The interactive evolutionary CFDG system displays each population of image-based phenotypes to allow users to select one or more parent designs by clicking on them with the mouse. The evolutionary CFDG system uses a standard one-point crossover operator and per-gene mutation to generate children from parents.

The evolutionary CFDG system was developed in Processing [6], a programming environment that the students were already familiar with. The code to

```

/* $RA = [-1,1] */
/* $XA = [-1,1] */
/* $YA = [-1,1] */
/* $RB = [-360,360] */
/* $SB = [0.85,0.99] */
/* $RC = [-1,1] */
/* $XC = [-1,1] */
/* $YC = [-1,1] */
/* $RD = [-360,360] */
/* $SD = [0.85,0.99] */

startshape LINES2
rule LINES2 {
    LINES1 { r $RA x $XA y $YA }
    LINES2 { r $RB s $SB }
}

rule LINES1 {
    LINE { r $RC x $XC y $YC }
    LINES1 { r $RD s $SD }
}

rule LINE { TRIANGLE { s 0.025 1 } }

```

(a) Template source code



(b) Renders

Fig. 2. A parameterised CFDG template with three rules and ten variables

the application was made available to the students allowing them to experiment with writing fitness functions to replace user selection. Some fitness functions such as surface area, X/Y symmetry, rotational symmetry, colour variance and brightness were provided as examples. Students created new fitness functions to guide the evolution of their parametric CFDGs.

3 Examples of Student Work

Examples of the outputs from evolutionary design grammars developed by students can be seen in Fig. 3a, giving some indication the breadth of possible design spaces. These examples were generated using the interactive evolutionary CFDG system. An example evolution of a design grammar using an image-based fitness function written by a student is given in Fig. 3b. The fitness function counts the number of colour changes by scanning horizontally across the centre of the image. This fitness function promotes complexity and in this example results in a design using tightly-packed, spiralling squares with alternating colours.

Using the extensions to Context Free, students were introduced to design grammars, parametric grammars, interactive evolutionary grammars, and finally non-interactive evolutionary grammars. This progression gave the students the opportunity to first learn about constructing design spaces before learning methods for searching them. All students demonstrated some understanding of how

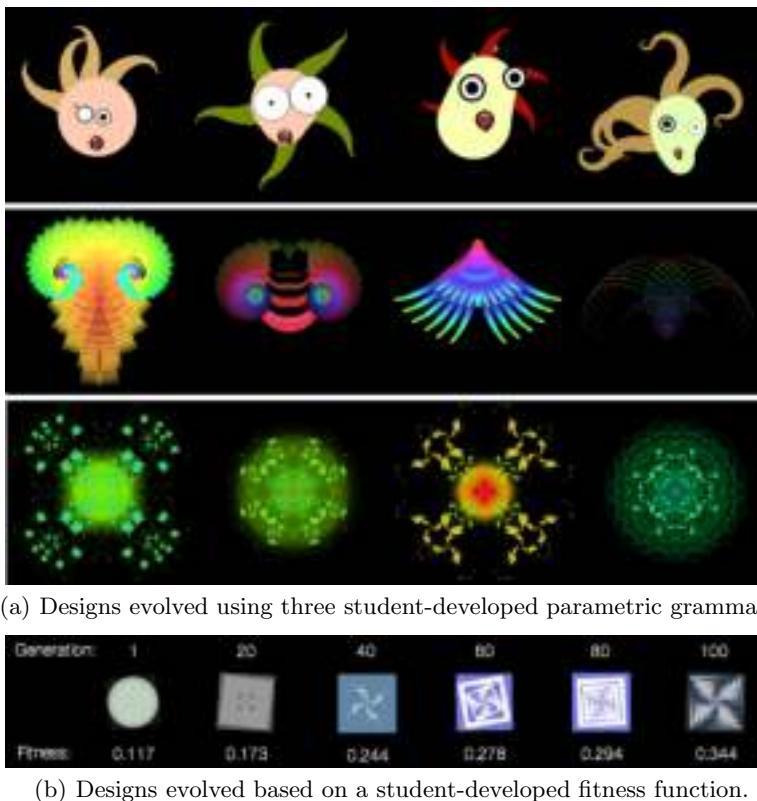


Fig. 3. Designs from (a) interactive and (b) non-interactive evolutionary systems

the definition of parameters as genotypes and rules as the means of expression worked together to define a design space.

Students were able to communicate the concepts they wanted to evolve using fitness functions, e.g., spikes, appendages, roundedness, gradients, pleasant colour combinations, produced documentation showing they possessed a good understanding of how GAs work and can be applied. We observed a varying degrees of success when they came to develop those concepts into algorithms. A minority of students were able to implement their fitness functions. Other students had difficulty following the implementations of the provided fitness functions but were able to apply and combine them to achieve results that they found interesting.

4 Discussion

The interactive evolutionary CFDG system proved an effective way to teach design students the principles of design grammars, parametric design and

evolutionary design. In contrast with previous teaching approaches, which involved students experimenting with configurable but fixed problems for each class of design system, the use of CFDG templates allowed students to implement their own problem domains very quickly.

The parametric template system, in which parameters can be inserted into any design grammar and interpreted as genes, allowed students to experiment with altering design spaces and immediately see how they affect the search process. This allowed students to experiment with the definition of their own genotypes, their own genotype-to-phenotype expressions and their own fitness functions. This allowed students to gain a deeper understanding of how the design of problems, representations and fitness functions affects the evolutionary design process. Students were able to experiment with the development of evolutionary systems with relatively little programming experience. Compared to other methods of teaching evolutionary design systems we found this method to be highly effective and enjoyable for the students to learn and experiment with.

Based on feedback from students, we believe our approach, based on a series of extensions to Context Free, shows great potential in the teaching of generative design systems to design students. Future iterations of this teaching method may benefit from an increased focus on the graphical user interface to the extended system. For example, to ease the implementation of fitness functions, students may benefit from the addition of a graphical interface to select and combine a set of existing fitness functions, as an alternative to writing code. This would further lower the exposure of the students to the underlying implementation while opening up the possibility for exploring the consequences of fitness function design to more students.

References

- [1] Glancey, J.: Secrets of the Birds Nest, The Guardian Online (last accessed January 2009), <http://www.guardian.co.uk/artanddesign/2008/feb/11/architecture.chinaarts2008>
- [2] Coyne, C.: Context Free Design Grammar (last accessed January 2009), <http://www.chriscoyne.com/cfdg/>
- [3] Horigan, J., Lentczner, M.: Context Free (last accessed January 2009), <http://www.contextfreeart.org/>
- [4] Lapré, L.: Lparser (last accessed January 2009), <http://members.ziggo.nl/laurens.lapre/lparser.html>
- [5] Lintermann, B., Deussen, O.: Xfrog (last accessed January 2009), <http://www.xfrog.com/>
- [6] Reas, C., Fry, B.: Processing: A Programming Handbook for Visual Designers and Artists. MIT Press, Cambridge (2007)

Artificial Nature: Immersive World Making

Graham Wakefield and Haru (Hyunkyung) Ji

Media Arts and Technology, University of California Santa Barbara,
Santa Barbara, California, CA 93106, USA
{wakefield,jiharu}@mat.ucsb.edu
<http://artificialnature.mat.ucsb.edu>

Abstract. Artificial Nature is a trans-disciplinary research project drawing upon bio-inspired system theories in the production of engaging immersive worlds as art installations. Embodied world making and immersion are identified as key components in an exploration of creative ecosystems toward art-as-it-could-be. A detailed account of the design of a successfully exhibited creative ecosystem is given in these terms, and open questions are outlined.

Keywords: Creative Ecosystems, Generative Art, Aesthetics, Bio-inspired, Art-as-it-could-be, Computational Sublime.

1 Introduction

Artificial Nature is a trans-disciplinary research project drawing upon bio-inspired system theories and an aesthetics of computational world making toward the production of immersive ecosystems as art installations (Fig. 1). Our motivation is to develop a deeper understanding of emergence and creativity as a form of art, study and play, by taking inspiration from nature's creativity but recognizing the potential of natural creation beyond the known and the physical.

Bio-inspired computational models such as evolutionary computation, multi-agent systems and computational development can be utilized in the construction

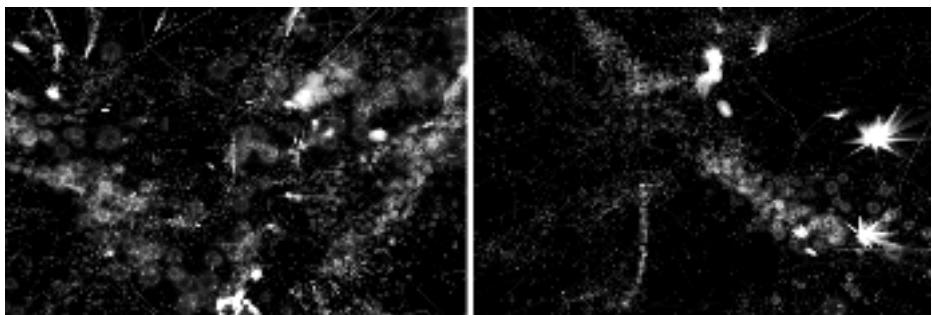


Fig. 1. Two screenshots from within the ecosystem

of artificial worlds with significant aesthetic potency [5]. From the perspective of ALife research, Etxeberria calls speculative or exploratory worlds *instantiations* in order to distinguish them from problem-solving *tools* or theory-producing *models*. Instantiations are more intuitively defined in terms of the ‘lifelike’, with a diffuse boundary between the natural and artificial; new creations of ontology that offer “the most radical ways of exploring life-as-it-could-be.” [6]

The suffix *-as-it-could-be* is central to our work: it indicates a shift in thought beyond the immediately apparent to the imaginable and possible: a shift which lies at the heart of exploratory creativity [13]. Indeed, Jon McCormack paraphrased Lagton’s *life-as-it-could-be*[8] into *art-as-it-could-be*[9]: an exploration of generative creativity not bounded by existent notions of aesthetics or artist.

For Artificial Nature we suggest a sensibility in which the viewer is embedded within the ecosystem itself, both through embodied interaction with complex systems, and by emphasizing an engaging, immersive aesthetic. The embodiment of an autonomous, emergent world allows us to explore and discover, rather than impose, its beauty. As such, we hope to dissolve the apparent tension in the conjunction of *nature* with *artificial*, the realization of creative *physis* through extended *poiesis*.

We can thus divide our project into a twofold demand: the construction of a world engendering emergent structures, and the design of an immersive mode of involvement in this world.

1.1 Exhibits

The installation has been exhibited at the Shanghai Exhibition Center, China (ASIAGRAPH, June 2008), Total Museum of Contemporary Arts, Seoul, Korea (thisAbility vs. disAbility, July-August 2008, Fig. 2), Seongnam Art Center, Seongnam, Korea (Universal Electronic Art, October-November 2008.) It is an ongoing installation (since January 2009) at the California Nano Systems Institute Allosphere[1], University of California Santa Barbara.

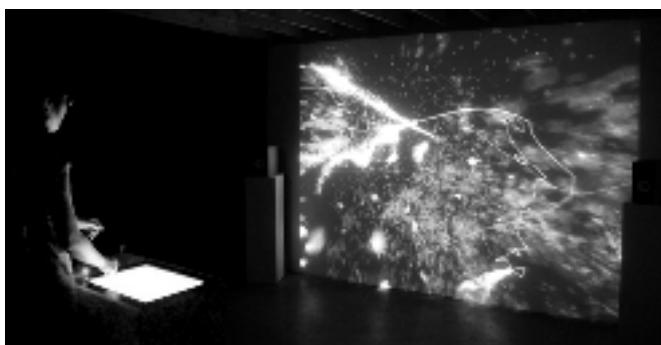


Fig. 2. Installation at the Total Museum of Contemporary Art, Seoul, Korea, 2008

2 Embodied World Making

The strategy of embodiment offers a means to preserve a sense of coherent materiality in the midst of unpredictable emergence. Embodiment asks that every structural observable have a systemic function within the world, and a genealogical or developmental account must be given as to how such structures and functions emerge from basic constituents (and how they disappear).

Ultimately the goal is not an aggregate of distinct models, but a single model with potential for many different kinds of creative development. In this respect we draw inspiration from Manuel De Landa's readings of Bergson and Deleuze [3,4], accounting for the production of apparent qualities as local emergent stabilities through progressive differentiations of an intensive virtuality into the multiply stratified.

This section covers progress toward this goal, in the form of a computational ecosystem, followed by an outline of the presently outstanding open questions.

2.1 The Current Ecosystem

At the heart of an ecosystem are animate organisms, or agents, however the importance of a dynamic supporting environment cannot be understated [14]. A self-organizing substrate can provide the building blocks of biological strata as well as a dynamic environment to spur endogenous evolutionary selection, in turn progressively determined by an increasingly autonomous biosphere.

Our ecosystem begins with a spatial field of mobile particles, coagulations of elementary pseudo-chemicals, which are continuously transported within a medium of fluid dynamics. Particles may react with one another, but matter/energy is preserved. The substrate is kinetic, thermodynamic, and dissipative [10]. Visually, particles have many advantages: individually they display chemical content (color), and collectively they indicate density and the turbulent flow.

We introduce animacy in the form of spatially situated concurrent processes with internal state (agents). Kinetically, agents may drift in the fluid currents or may also use stored energy to trigger autonomously directed movement (influencing the fluid flow). Energetically, agents must constantly exchange elements with their local field for growth and behavior, and discharge toxic waste to prevent decay (metabolism/autopoiesis); however an organism with sufficient energy storage may also reproduce by binary fission.

Agent growth follows a developmental pattern by *gradually* evaluating genome data: arbitrarily structured graphs of elements that are converted piecewise into executable functions. Functions respond to organism state, including form, age, energy storage, and perceptions of the local environment etc., to produce behavior of movement, reorientation, consumption, growth, fission and genetic transfer.

The genes themselves evolve independently of reproduction through lateral gene transfer with other agents in close proximity. Agents may sing their genome and adopt the genome of a song heard, while mutation occurs through sonic imperfections. Selective pressures thus emerge endogenously through the complex feedback between gene expression, organism behavior and abiotic conditions

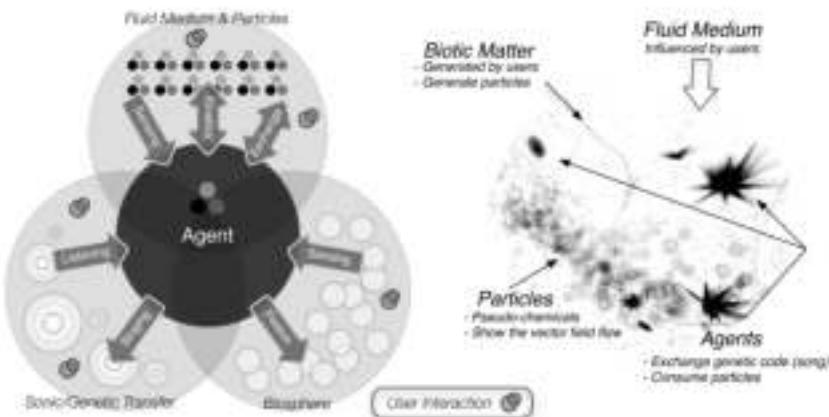


Fig. 3. Elementary components within the Artificial Nature ecosystem

(Fig. 3), transcending selective optimization problems of a priori fitness or the human bottleneck [9].

The installation software was developed using LuaAV [11] with extensions based upon [12] for fluid dynamics and [2] for organism geometry.

2.2 Open Questions

Our current research endeavors to flesh out details of how undifferentiated points of autonomy develop into differentiated, constrained organisms in mutual feedback with the environment [7] from elementary potentials that tie form to function. To follow the strategy of embodiment, the distinction between agent and environment ought not to be absolute: the process by which the animate (and likewise genetic material) emerges from the inanimate should be defined. This initial emergence remains an open research question.

At the macro-scale, we are interested in developing modes of agent communication that engender emergent behavior, such as territory marking or collaboration. Again, an account must first be made as to how such a capacity emerges in an agent.

3 Immersion

Throughout the design of the installation, we take into account the principle of the human as part of and embedded within the artificial ecosystem, both objectively in the form of mutual systemic interactions, and subjectively by heightening the sensation of immersion.

The viewer is situated locally within the 3D world rather than viewing from outside the system; the exhibition space supports aural immersion by surrounding the visitor with the background sounds of the world and songs of the agents

(the 360-degree stereographics and spatial audio to be available in the Allosphere [1] later in 2009 will greatly strengthen this impression.)

The design avoids easy visual referents to the actual world, supporting the exploratory context. For example, the sense of scale relative to the real is ambiguous (it may suggest star systems as much as a pre-biotic soup).

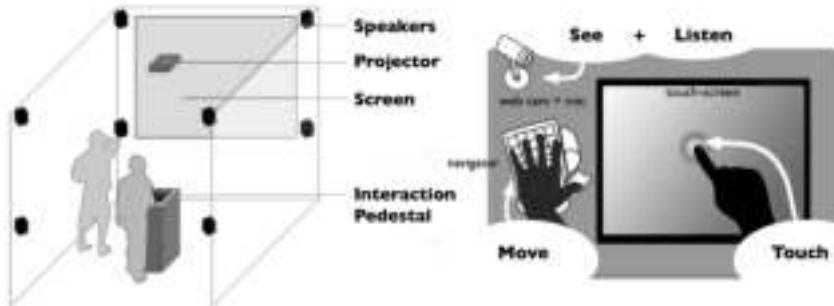


Fig. 4. Schematic of the installation, with interaction pedestal detail

3.1 Interactions

The visitor may interact with the world wilfully by means of a physical navigator device and touchscreen, mounted on a central pedestal (Fig. 4). Navigation (in six degrees of freedom) permits a literally exploratory mode of interaction, while immersion is intensified by the mutual influence of the fluid medium and the navigating user. The touchscreen can be used to add biotic matter to the local environment. By using the touchscreen and navigator together complex spatial distributions are possible.

In addition, involuntary activity of the visitor is detected through a camera and microphone, adding turbulence to the local fields. The world persists without need of human interaction, but responds in rich ways to it. It is very difficult for human interaction to have any intentional selective impact upon the ecosystem; such impact may only emerge through the complex detail of chain reactions.

3.2 Responses

Responses to the installation have been rich and varied, and many viewers have remained engaged with the work for extended periods of time. We have been pleased to discover that children are particularly fascinated by the work. Several viewers have commented on a sense of ‘being present’, or engaging with another life, regardless of their familiarity with the complex systems involved.

4 Conclusion

Artificial Nature provokes speculation on the concepts of creativity and beauty in both nature and culture. The evolving beauty of emergent complexity in our

project is intrinsically man-made yet follows an understanding of the mechanisms of nature itself: both cultural and natural worlds create and are formed by information flow through the traces of their own becoming.

Artificial Nature is not yet emergent to the extent ultimately desired, however it is unpredictable in detail while relatively stable in general. By making the systemic requirements and processes explicit through our development of the project, and following a strategy of embodiment, the next steps to be taken have been clearly revealed.

Acknowledgments. With gratitude to the Allosphere and the Interdisciplinary Humanities Center, University of California Santa Barbara, for financial support.

References

1. California nano systems institute allosphere,
<http://www.mat.ucsb.edu/allosphere/>
2. Bourke, P.: Geometry, surface, curves, polyhedra,
<http://local.wasp.uwa.edu/~pbourke/geometry/>
3. De Landa, M.: A Thousand Years of Nonlinear History. Zone Books (2000)
4. De Landa, M.: Intensive Science & Virtual Philosophy. Continuum International Publishing Group (2005)
5. Dorin, A.: A survey of virtual ecosystems in generative electronic art. In: Romero, J., Machado, P. (eds.) *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, pp. 289–309. Springer, Heidelberg (2007)
6. Etxeberria, A.: Artificial evolution and lifelike creativity. *Leonardo* 35(3), 275–281 (2002)
7. Kumar, S., Bentley, P.: *On Growth, Form and Computers*. Academic Press, London (2003)
8. Langton, C.J.: *Artificial Life: An Overview*. MIT Press, Cambridge (1995)
9. McCormack, J.: Open problems in evolutionary music and art. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2005*. LNCS, vol. 3449, pp. 428–436. Springer, Heidelberg (2005)
10. Prigogine, I., Stengers, I.: *Order Out of Chaos*. Bantam (1984)
11. Smith, W., Wakefield, G.: Computational audiovisual composition using lua. In: Adams, R., Gibson, S., Arisona, S.M. (eds.) *Transdisciplinary Digital Art. Sound, Vision and the New Screen*, pp. 213–228. Springer, Heidelberg (2008)
12. Stam, J.: Real-time fluid dynamics for games. In: Proceedings of the Game Developer Conference (January 2003)
13. Sternberg, R.J.: *Handbook of Creativity*. Cambridge University Press, Cambridge (1998)
14. Weyns, D., Schumacher, M., Ricci, A., Viroli, M., Holvoet, T.: Environments in multiagent systems. *The Knowledge Engineering Review* 20(02), 127–141 (2005)

Evolving Indirectly Represented Melodies with Corpus-Based Fitness Evaluation

Jacek Wolkowicz, Malcolm Heywood, and Vlado Keselj

Faculty of Computer Science, Dalhousie University, 6050 University Ave.,
B3H 1W5 Halifax NS, Canada
{jacek,mheywood,vlado}@cs.dal.ca

Abstract. The paper addresses the issue of automatic generation of music excerpts. The character of the problem makes it suitable for various kinds of evolutionary computation algorithms. We introduce a special method of indirect melodic representation that allows simple application of standard search operators like crossover and mutation with no repair mechanisms necessary. A method is proposed for automatic evaluation of melodies based upon a corpus of manually coded examples, such as classical music op. Various kinds of Genetic Algorithm (GA) were tested against this e.g., generational GAs and steady-state GAs. The results show the ability of the method for further applications in the domain of automatic music composition.

Keywords: Music generation, unigrams, MIDI, generational genetic algorithms, Steady-state genetic algorithms, automatic fitness assessment.

1 Introduction

Music is a well-structured organization of notes, thus one can probably design algorithms and systems to support the various processes of music composition. Applications for music touch many fields of computer science, but evolutionary computation seems to be especially suited for music; not least because music melodies have a linear, sequential nature, which clearly lends itself to representation in terms of a gene structure. Moving beyond representational issues, there are still many outstanding problems that are worthy of study. A few major problems to address were summarized by McCormack [6]. This paper proposes approaches to two of them: the problem of creating a music representation especially suited for evolutional techniques and developing an automated fitness function that is able to select pleasant individuals.

Both the most important and the toughest one is the evaluation of melodies in terms of their fitness. A judgment whether certain melody sounds ‘good’ or ‘bad’ is a vague issue even for humans. Many solutions to this problem try to overcome this problem by manual fitness assignment, but this compromises our ability to automate the process of music creation. Other approaches include fitness-assigning methods that result from some theoretical models of music. However, limiting melody creation results in artificial constraints on the resulting composition. The third approach, which

will be used in this paper, utilizes sample pieces from a corpus to create classifiers able to judge the properties inherent in new melodies.

2 Previous Work

In early approaches, introduced by Biles [1] or Jacob [3], the system relied on human judgment as feedback for fitness evaluation. Johanson and Poli [4] augmented the method with a neural network that used human feedback from their previous experiments to automatic evaluation of later melodies.

Papadopoulos and Geraint [7] pointed out that there is no formal approach for establishing the automatic evaluation of music quality, proposing a method for automatic fitness evaluation based on static melody features. Other approaches to the problem of automatic fitness assignment involve Neural Networks [9], SOMs [8], fixed musicological rules [11] or similarity to a target melody [2]. The most similar approach to the one proposed in this paper is based on Zipf's law for music pieces [5].

3 Methodology

The approach presented in this paper is a novel approach to representing melodies and in so doing avoids utilizing human feedback during fitness evaluation. However, a sample of MIDI files is necessary to provide the source from which music features are extracted. Specifically, a complete set of Preludes and Fugues from Das Wohltemperierte Klavier 1 by J.S. Bach establishes the source of music features, although no special significance is associated with this selection.

The system operates on the Western music twelve-tone scale. However, unlike previous approaches, where notes were represented using absolute note pitches and timings [2], here the smallest melody component will be the information about pitch change and duration change (as proposed in [10]). The genome is a sequence of genes, dubbed unigrams – pairs of integer values indicating the interval between notes in semitones and the rounded binary logarithm of the ratio of the corresponding note's duration. Figure 1 shows the process of obtaining unigrams from a simple melody.

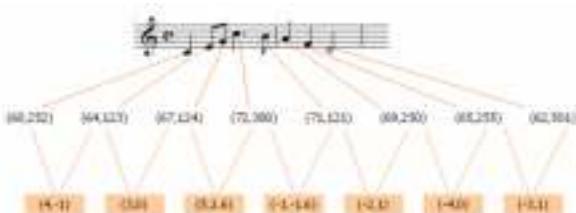


Fig. 1. Unigram extraction process

Naturally, the same melody can be played in different scales or/and different tempos and it still remains the same melody and preserves relations between notes. By emphasizing the underlying relation between notes we fulfill the human

perception of rhythm and melody. Since the pitch and the tempo of a melody is not determined or needed, the scale and the tempo are necessary for playback only.

The other advantage of this approach over direct representations is that the logic of the melody is preserved. Moreover, application of classical one/two point crossover does not result in lethals (Figure 2).

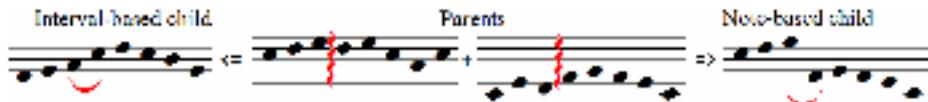


Fig. 2. Two children created using different representations for standard crossover

3.1 GA Design Decisions

The evaluation of fitness is automatic and reflects the likelihood that the given melody is a good melody given a corpus of musical compositions. The features of the corpus are calculated by counting all melodies of ‘ n ’ consecutive unigrams, i.e., n -grams, and taking statistics of n -grams in the corpus like n-gram frequencies or document frequencies (the count of documents in which the n -gram occurs). Then, fitness can be evaluated according to the following general fitness function:

$$Fitness(melody) = \Gamma_1 \left(\prod_{1 \leq n \leq N} w_n \cdot \Gamma_2 \sum_{n\text{-gram} \in melody} \varphi(n\text{-gram}) \right) \quad (1)$$

where Γ_1 and Γ_2 are some aggregate functions e.g., sum, average, max; Γ_1 has the range over the various sizes of n and Γ_2 has the range of all n -grams in an individual with the size of n ; w_n is a weighting factor of certain n -gram size; $\varphi(n\text{-gram})$ is a function of a certain n -gram that is derived from corpus statistics.

Two forms of selection operator will be considered: Standard generational (roulette-wheel) and a steady-state tournament. In the case of search operators, the representation presented in this paper allows several classical operators to be supported directly without the need

One-point crossover is defined such that the crossover point is selected independently in each individual. That is to say, a melody coded using the method proposed in this paper does not have the limitation where a gene at a position has a certain meaning according to its localization in the genome. Moreover, we do not want to impose a priori limits on the evolution of melody length.

Mutation consists of replacing a gene with a new tuple. Since unigrams represent a change of melody, drawing a tuple from the corpus probabilistically will change a single inflection point of the melody leaving the rest of the melody (relatively) unaffected.

4 Experiments

The following experiments were conducted to characterize various features of the problem. The population consists of 100 individuals, each being a sequence of 10 unigrams drawn randomly according to the probabilities obtained from the corpus. All

experiments were written as Perl scripts using Perl::Midi and Perl::Midi::Corpus libraries for analyzing and managing *n*-gram features of MIDI files and individuals in the populations.

The key point in this system is to define the details of the fitness function. The simple and most obvious decision is using sum (or average) for Γ_1 and Γ_2 . One will then have ranked individuals according to the average fitness of their components. The main challenge is how to evaluate n-grams (individual sub-sequences) for their likelihood of being good or bad melodies. The simplest approach is to take the probability of n-grams. However, it transpired that despite the fact that ‘good’ melodies are usually quite complicated, the majority of the corpus is built from very simple elements. The melodies evolved from φ functions based on the probability model resulted in passages of notes just going up, down or simple trills.

The simple and conceptually acceptable solution to this problem is to use a variant of the tf-idf measure from Information Retrieval, where it determines term importance. Most important terms are the ones that are frequent, but occur in few documents. The following formula was used to express the usefulness of an *n*-gram:

$$\varphi(n\text{-gram}) = \frac{\log(tf(n\text{-gram}))}{df(n\text{-gram})} \quad (2)$$

4.1 Generational GA

Having established a suitably informative fitness function, we are in a position to apply a standard Generational GA (GGA). The fitness function represented the sum (Γ_1) of five averages (Γ_2) associated with component *n*-grams (*n* from 2 to 6). The average values for fitness was around 1.5 and maximum fitness was changing rapidly reaching the level of 3 (average φ value of 0.6) without the ability of keeping these good genes across generations. It was still far from the peak φ values (~ 4.0).

In later generations a new phenomenon occurs – ‘note bloat’, unlimited growth of individuals, similar to code bloat known from GP. It may result from many factors. The individuals are not limited in length. Crossover is known to have a destructive influence on the offspring. The influence of a wound after crossing over some ‘valuable’ regions may be hidden by the length of individuals. Hence shorter children are more likely to suffer from this destructive influence than longer ones. Populations with larger individuals register lower fitness properties (average and maximal).

4.2 Steady-State GA

A Steady-state, elitist selection operator always maintains the best solutions. Only one child is created per iteration and the worst solution is replaced. An illustrative sample run of the system in this approach is shown in Figure 3a.

Unlike GGA, the variety in the population drops and the whole population ‘converges’ to a single individual. There is no note bloat and the quality of solutions is higher compared to those identified by GGA. Figure 3b shows the average behaviour over 100 runs. In the first 5-10 runs the algorithm searches for a niche to converge (increasing size of individuals), thereafter, a single solution appears.

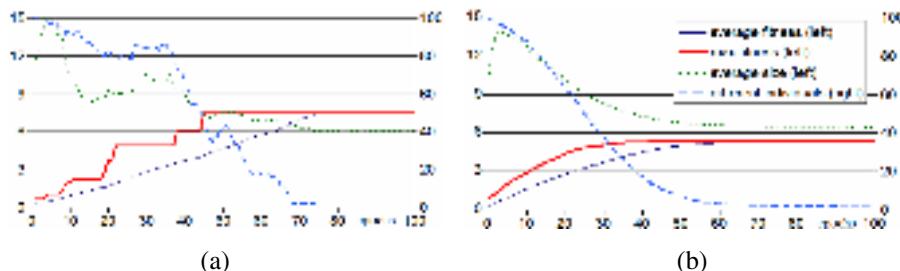


Fig. 3. (a) Steady-state GA run over 100 epochs. (b) Average results of 100 runs of the system.

Mutation can inject new genes, resulting in new component melodies, which may be more or less fit than those currently in the population. Steady-state selection insulates us from the destructive influences of mutation. Moreover, it appears that mutation is important; the rate of 0.2 results in a fitness improvement of around 20%.

Table 1. The Sample of resulting melodies

fit.	pattern	fit.	pattern	fit.	pattern
9.55		8.37		7.98	
8.83		8.21		7.48	
8.46		8.05		7.17	

4.3 Results

The melodies generated by the system are shown in Table 1. Each melody is preceded by its fitness. They were taken as final results from 100 runs of the steady-state algorithm. It is usually easy to put those melodies into a harmonic context and most of them preserve the scale or the key in which they are being played. Most of them are in major scale, but some of them are complementary: the melody with the fitness of 8.21 is the major version of the minor melody with fitness 7.98.

5 Conclusions and Future Work

The method for generating music melodies using a corpus of musical pieces as a guide for evaluating individuals gives encouraging results without resorting to sophisticated evolutionary frameworks. Applying some knowledge of music while formulating the representation has provided the basis for an indirect encoding, thus avoiding the need for specialist search operators. The problem of evolving melodies also demonstrates some new artefacts such as ‘note bloat’ that may be very interesting to investigate.

The results of the approach to date are characterized by a quite simple rhythmic scheme. Separating rhythm from melody may help in obtaining more diversified results, but one cannot make rhythm and melody totally independent from each other. Other music features like establishing a proper beginning and ending and maintaining a common harmonic logic and structure of an excerpt also need to be investigated. The evaluation of results is still an open-ended issue.

References

1. Biles, J.: GenJam: A Genetic Algorithm for Generating Jazz Solos. In: Proceedings of the 1994 International Computer Music Conference, Aarhus, Denmark (1994)
2. Gartland-Jones, A.: MusicBlox: A Real-Time Algorithmic Composition System Incorporating a Distributed Interactive Genetic Algorithm. In: Raidl, G.R., et al. (eds.) *EvoMusArt 2003*. LNCS, vol. 2611, pp. 490–501. Springer, Heidelberg (2003)
3. Jacob, B.L.: Composing with genetic algorithms. In: Proceedings of the 1995 International Computer Music Conference, pp. 452–455 (1995)
4. Johanson, B., Poli, R.: GP-Music: An interactive genetic programming System for music generation with automated fitness raters (Technical report CSRP-98-13). School of Computer Science, The University of Birmingham, Birmingham, UK (1998)
5. Manaris, B., Machado, P., McCauley, C., Romero, J., Krehbiel, D.: Developing fitness functions for pleasant music: Zipf's law and interactive evolution systems. In: Rothlauf, F., et al. (eds.) *EvoMusArt 2005*. LNCS, vol. 3449, pp. 498–507. Springer, Heidelberg (2005)
6. McCormack, J.: Open Problems in Evolutionary Music and Art. In: Rothlauf, F., et al. (eds.) *EvoMusArt 2005*. LNCS, vol. 3449, pp. 428–436. Springer, Heidelberg (2005)
7. Papadopoulos, G., Wiggins, G.: A Genetic Algorithm for the Generation of Jazz Melodies. In: Proceedings of the Finnish Conference on Artificial Intelligence (SteP 1998), Jyvaskyla, Finland, September 7-9 (1998)
8. Phon-Amnuaisuk, S., Law, E.H.H., Kuan, H.C.: Evolving Music Generation with SOM-Fitness Genetic Programming. In: Giacobini, M., et al. (eds.) *EvoMusArt 2007*. LNCS, vol. 4448, pp. 557–566. Springer, Heidelberg (2007)
9. Oliwa, T., Wagner, M.: Composing Music with Neural Networks and Probabilistic Finite-State Machines. In: Giacobini, M., et al. (eds.) *EvoMusArt 2008*. LNCS, vol. 4974, pp. 503–508. Springer, Heidelberg (2008)
10. Wolkowicz, J.: N-gram-based approach to automatic composer recognition. Master's Thesis. Warsaw University of Technology (2007)
11. Worth, P., Stepney, S.: Growing Music: Musical Interpretations of L-Systems. In: Rothlauf, F., et al. (eds.) *EvoMusArt 2005*. LNCS, vol. 3449, pp. 545–550. Springer, Heidelberg (2005)

Hearing Thinking*

Jane Grant, John Matthias, Tim Hodgson, and Eduardo Miranda

Interdisciplinary Centre for Computer Music Research (ICCMR)

Faculty of Arts and Faculty of Technology, University of Plymouth,

Drake Circus, Plymouth PL4 8AA

j1grant@plymouth.ac.uk, john.matthias@plymouth.ac.uk,
tim.hodgson@plymouth.ac.uk, eduardo.miranda@plymouth.ac.uk

Abstract. This paper describes early experiments, which attempt to reconfigure the sound of a breath using a network of artificial spiking cortical neurons. The connectivity of the network evolves according to a spike timing dependent plasticity algorithm and the instrument triggers grains of sound from recordings of the breath when any of the neurons fire.

Keywords: Cortical Neurons, Granular Synthesis, Synaptic Plasticity.

1 Introduction

When a person sees, light enters the eyes. The information is translated into electrical signals via sensory neurons and is processed in the brain via cortical neurons. The information is then interpreted by the brain, and any action taken will be via motor neurons which will result in external motor action.

When a person hears, sound enters the ears. The information is translated into electrical signals via sensory neurons and is processed in the brain via cortical neurons. The information is interpreted by the brain, and any action taken will be via motor neurons which will result in external motor action.

In terms of vision, we can define the process involving the seeing, the sensory and cortical processing and the interpretation as ‘looking’. In terms of audition, we can define the sensory and cortical processing and the interpretation as ‘listening’

This project began with a specific interest in the relationship between our sensual perception of the external world (for example, looking and hearing) and the externalization of that perception in relation to the idea of ‘self’. The specific externalization (action) focused on is the voice and more precisely the breath, the beginning of vocalization. Daniel C. Dennett [1] proposes the notion of ‘self’ as a construction; that ‘the self’ is a ‘naive boundary between “me” and “the outside world”’ and suggests that ‘even such a simple self is not a concrete thing but just an abstraction, a principle of organisation. Moreover the boundaries of a biological self are porous and indefinite.’ In the field of phenomenology, David Wood [2] argues that such boundaries are interchangeable as states, that ‘a boundary is not a thing, but

* This paper was accepted for publication at EvoMUSART 2008. Unfortunately, it wasn’t published due to a communication problem.

a cluster of procedures for the management of otherness'. The central artistic aim is to affect a rupturing of the boundaries between the sensed and the action and explore the effects of removing the sensing and sensory part of the 'self'. In order to examine the effect of such a rupturing, we have merged an externalised action (the recorded sound of a breath) with a sonic instrument, which creates sound from a model of cortical neuronal firing patterns known as the Neurogranular Sampler [3].

The Neurogranular sampler works by triggering grains of sound (typically in a range of duration of 10 milli seconds (ms) -50ms) taken from a recorded sample when any one of an isolated network of artificial cortical neurons 'fires'. The resulting sound therefore consists of short bursts of the original sample triggered by the cortical neurons. It is a sonification of the cortical firing patterns. In this work, the sound of both voice and breath are recorded then reconfigured via the Neurogranular sampler in order to merge the voice or breath with the patterns and rhythms occurring in the neuronal network; we 'hear' neurons firing but within the recognised 'sound frame' of the human breath. The voice, but more particularly the breath itself belongs to an integral internal to external process and is one that we all recognise readily. There is a transition from internal to external, a process which might result in the listener hearing both the alien and the familiar simultaneously, existing at a liminal threshold.

2 The Neurogranular Sampler

The Neurogranular sampler [3] utilizes the recently developed model of a spiking neural network of Izhikevich et. al. [4]. The methodology of sonification we use for the Izhikevich model uses the time of firing as the main sound event. All of the parameters in the model are used as controllers for the time of firing for each neuron. Every time a neuron fires, a dot is placed in a sound event file. We use these events as temporal points at which sounds are generated. The dynamics of the spiking network are determined and controlled by varying the number/type of neurons and the geometry of the connectivity. In the original work of Miranda and Matthias [3], the elements of the matrix of connections, S were updated using the simple Izhikevich algorithm [4]. In the simple algorithm, elements of S are fixed and added to the postsynaptic membrane potential instantaneously following a firing event. In the current implementation, the elements of the matrix S of connections are updated either using the simple model or according to a Spike Timing Dependent Plasticity (STDP) algorithm [5]. In this algorithm, connections for which pre-synaptic spikes cause post-synaptic firing are potentiated and those for which pre-synaptic spikes arrive after post-synaptic firing has occurred are depressed. We also have included axonal conduction delays [6], which are random numbers between 1ms and the maximal conduction delay parameter, which in our implementation is of the order of 10-20ms.

The neuronal network remains isolated to any sensory information and is unstimulated from outside the network. It is driven by noise such that the initial current and voltage parameters within a proportion of neurons will be of a high enough value to drive them to fire. Figure 1 shows a typical raster plot in a simulation of 1000 neurons. We can see that the firing response is highly correlated and that the firing forms a coherent pulse across the network.

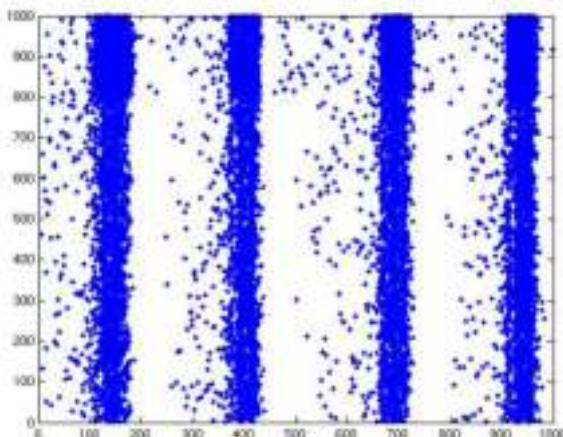


Fig. 1. A graph showing neuron number (y-axis) plotted against time (milli seconds, x-axis) in a simulation using the Izhikevich model including spike timing dependent plasticity and axonal conduction delays. The maximum axonal conduction delay parameter in this simulation is 20ms.

2.1 Sonification

In our methodology, short segments (or “sound grains”) taken from sound files are triggered when any of the neurons fire. The algorithm by which the system works is straightforward: When a neuron in the network fires at time t , a sound grain of predetermined length (between 10-50ms) and amplitude is taken from the recorded sample of sound and played back. Synchronized firing of many neurons sound like a pulse, whilst networks containing only a few neurons produce sparse rhythmic sequences. The system therefore has a very wide variety of temporal patterns and behaviours, which can be controlled according to the parameters of the model. One can control the parameters which determine the intrinsic properties of the neurons [4] and one can control the number and type of neurons. Generally speaking, increasing the number of neurons in the model means more firing and therefore more sonic texture, although when the solutions exhibit synchronous behaviour, increasing the number of neurons tends to lower the frequency of the collective response. It is interesting in itself that such random (noisy) inputs can produce synchronous rhythms, a well understood phenomenon within the dynamical systems community. Figure 2 shows the output amplitude from the Neurogranular Sampler against time in an early simulation which consists of 4 excitatory Neurons and 1 inhibitory neuron with an axonal delay of up to 10ms including STDP.

The STDP algorithm along with the axonal delays encourage particular pathways in the network to become established, which lead to more events and more regular frequency of neuronal firing. In the absence of sensory input to the network, these pathways have a transient lifetime.

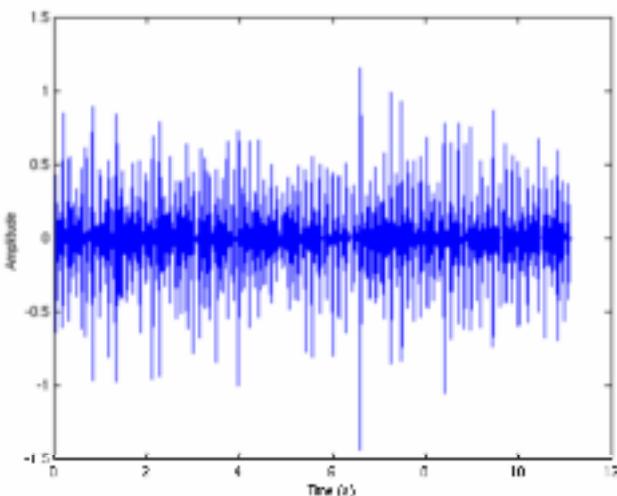


Fig. 2. A simulation of the Neurogranular Sampler, which shows the amplitude of signal plotted against time in seconds for an artificial network using the Izhikevich model with 4 excitatory neurons and 1 inhibitory neuron all of regular spiking type including an axonal delay of up to 10ms and Spike-timing dependent plasticity. The initial voltage and current parameters were taken as random matrices. The external sound in this example is taken from a single note played on a harmonium.

3 Methodology

The process of experimentation for this work initially consisted of designing a user interface for the Neurogranular Sampler. This was written in MATLAB and enabled the user to choose the number of neurons in the network, individual neuronal firing characteristics, an initial matrix for the initial current input to each neuron and the sound file for sampling. The user has a choice whether to implement a dynamic matrix of connections S via the STDP rule, or to use the simple static initially configured matrix of connections. Experiments with the interface consisted of inputting a sequence of breaths and vocal sounds into the sampler. The parameters on the interface were chosen in order to achieve a rupturing effect such that the sound of the (external) breath is reconfigured to enable the listener to hear the external reconfigured within the context of the (internal) firing.

The initial experiments of Miranda and Matthias [3] use very short grains, the result being a direct rhythmical translation of the simple neuronal firing patterns. In addition, the effect of using very short grains renders the result rather decoupled from the initial recording. In this work, in order to affect the rupturing between the breath and the processing, we used much longer grains than had previously been used. In fact the possible duration of grain sizes has been greatly increased, to a point where they cannot really be considered 'grains' any more. The resultant layering of multiple samples gives rise to some intriguing sonic effects: When using the larger grain lengths, we experienced a real lengthening and spatial aspect to the experimental

outcomes. We also decided to depart from the original sampler design, in which grains were taken at random from any point in the input sound file. We felt that this removed too much of the structural character from the source material. Instead, the user can define a 'grain window', limiting how far in time from its original position any given grain can move. This may be anything from 1ms to the entire length of the output file, giving the user great flexibility in determining how radically the Neurogranular sampler transforms the input. In this implementation, the breath or voice appears to be drawn out, but simultaneously held together by a series of 'strands' of information. These strands merge to form a singular long breath ruptured at times by waves winding through the sample. Figure 3 shows the output from a simulation from a recorded breath using a neuronal network including 6 excitatory and 6 inhibitory neurons. The grain size was allowed to vary between 695 and 4352 samples for which the sampling rate was 44.1kHz.

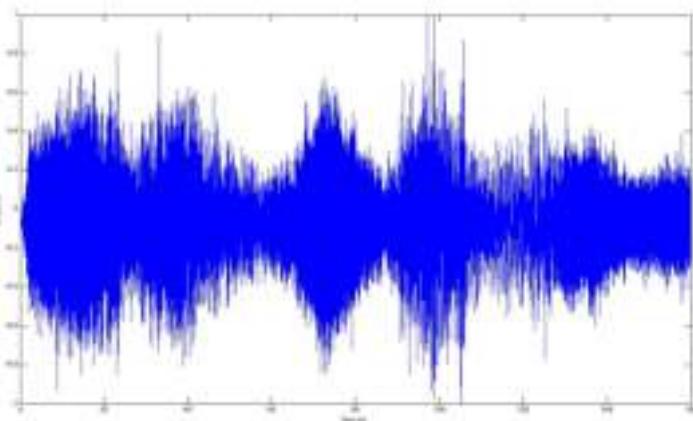


Fig. 3. A simulation of the Neurogranular Sampler, which shows the amplitude of signal plotted against time in seconds for an artificial network using the Izhikevich model with 6 excitatory neurons and 6 inhibitory neurons all of regular spiking type including an axonal delay of up to 20ms and Spike-timing dependent plasticity. The recorded sound was a breath, which lasted for approximately 10seconds. The grain size was allowed to vary in duration between 695 and 4352 samples. The sampling rate was 44.1kHz.

4 Discussion

In this work, there is no sensory information input to the network of cortical neurons. The 'sensory' information exists in the recorded sample, which is retriggered and reconfigured according to the firing patterns of an isolated tiny cortex. There is no 'hearing' or 'seeing' but a reconfiguring of a sound already heard. The sounds emulate the strangeness of what we might imagine the merging of the voice with a neural structure could sound like, the more interesting results causing a tension between what we perceive to be an internalized sound but resonating with an external spatial aspect. In further work, we intend to explore how the system might respond

when sensory information is directly input to the neuronal network, integrating the sensory/cortical interaction. We also intend to explore the resulting dynamic evolution of the synaptic connections and firing pathways in the light of this sensory input, and additionally, produce a more interactive environment in which the user can work with the network in real time, enabling them to adapt the input according to the results of the processing.

As discussed at the beginning of this paper, one of the starting points in this work is the philosophical premise that the idea of ‘self’ is a construction, that ‘the self’ has no concrete boundaries. Daniel C. Dennett argues [1] that the boundaries of a biological self are ‘porous and indefinite’. The experiments outlined in this paper will eventually form the audio part of a sound and video installation called ‘Threshold’, by Jane Grant. This work is concerned ideas outlined here and also with the sense of space both infinitesimal and infinite, which the breath or voice inhabits. Gaston Bachelard in his book, ‘The Poetics of Space’ [8] suggests that there lies an infinity within the pronunciation of the word ‘vast’, that in saying the word it ‘teaches us to breathe with the air that rests on the horizon...’

The final work for this installation will send waves of sound undoing the metaphorical architecture of the brain and dispersing it towards the horizons.

References

1. Dennett, D.C.: *Consciousness Explained*, Penguin (1991)
2. Wood, D.: Time shelters: an essay in the poetics of time. In: Durie, R. (ed.) *Time and the Instant: essays in the physics and philosophy of time*, Clinamen Press, Manchester (2000)
3. Miranda, E.R., Matthias, J.R.: Granular sampling using a pulse-coupled network of spiking neurons. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) *EvoWorkshops 2005. LNCS*, vol. 3449, pp. 539–544. Springer, Heidelberg (2005)
4. Izhikevich, E.M.: A simple model of a spiking neuron. *IEEE Transactions on Neural Networks* 14, 1469
5. Song, S., Miller, K.D., Abbott, L.F.: Competitive Hebbian Learning through spike-timing dependent synaptic plasticity. *Nat Neuroscience* 3, 919–926 (2002)
6. Izhikevich, E.M., Gally, J.A., Edelman, G.M.: Spike Timing Dynamics of Neuronal Groups. *Cerebral Cortex* 14, 933–944 (2004)
7. Abramovitz, M., Stegun, I.: *Handbook of Mathematical Functions*. Dover, New York (1972)
8. Bachelard, G.: *The Poetics of Space*. Beacon Press (1964)

Memetic Variation Local Search vs. Life-Time Learning in Electrical Impedance Tomography

Jyri Leskinen, Ferrante Neri, and Pekka Neittaanmäki

Department of Mathematical Information Technology, Agora,
University of Jyväskylä, P.O. Box 35 (Agora),
FI-40014 University of Jyväskylä, Finland
{jyril,neferran,pn}@jyu.fi

Abstract. In this article, various metaheuristics for a numerical optimization problem with application to Electric Impedance Tomography are tested and compared. The experimental setup is composed of a real valued Genetic Algorithm, the Differential Evolution, a self adaptive Differential Evolution recently proposed in literature, and two novel Memetic Algorithms designed for the problem under study. The two proposed algorithms employ different algorithmic philosophies in the field of Memetic Computing. The first algorithm integrates a local search into the operations of the offspring generation, while the second algorithm applies a local search to individuals already generated in the spirit of life-time learning. Numerical results show that the fitness landscape and difficulty of the optimization problem heavily depends on the geometrical configuration, as well the proposed Memetic Algorithms seem to be more promising when the geometrical conditions make the problem harder to solve.

1 Introduction

The Electrical Impedance Tomography (EIT) is a cost-effective and robust technique which allows to study the internal structure of an object without physically opening it [1]. This technique can be applied to various fields of applied science, especially in medicine where the necessity of analyzing object structures (e.g. analyzing the presence of anomalies in a human head) without an invasive approach is crucial.

The EIT is implemented by inserting a set of electrodes, distributed on the surface of the object, and then injecting a small current through some of these electrodes while the remaining electrodes are employed in order to measure variations in the voltages [2]. The same experiment is performed under various voltage configurations and several measurements are then recorded. These measurements are used to compute (since the current is known) the internal resistivity distribution. The resulting image allows a mapping of the object on the basis of resistivity values and thus detection of internal objects (e.g. a brain cancer) characterized by resistivity values different from the background.

Mathematically speaking, the EIT is a non-linear inverse problem which is severely ill-posed [3] i.e. the problem does not respect the Hadamard well-posedness criteria of existence and uniqueness of the solution, and continuous

dependence of this solution on the data. The mathematical formulation details of the EIT inverse problem are given in [4] and [5]. In a nutshell, the EIT inverse problem can be seen as detection of the resistivity values internal to the object under study, after that a mapping of the internal area by means of a mesh has been performed. These resistivity values and the given values of injecting currents allow a simulation of the electric structure, and thus, detection of the voltage values on the object surface, in correspondence to the located electrodes. The subsequent problem, which is the focus of this paper, is detection of that set of resistivity values which ensure a minimal difference between the simulated and measured voltage values, respectively.

This optimization problem has been studied in literature and several approaches have been tested. The most popular approaches propose the application of Newtonian and quasi-Newtonian methods, see for example [2] and [6]. The main advantage with the application of these methods is that they require a limited amount of iterations to converge to a solution. On the other hand, these approaches have a grave drawback in that they perform a search in the decision space by imposing a continuous variation of resistivity values. This fact leads to a smoothing effect on the resistivity values in the resulting image which obviously does not allow a proper identification of the internal object in either its shape or in its resistivity values, but only gives some indication on its presence and location. In other words, Newtonian and quasi-Newtonian methods implicitly modify the EIT inverse problem.

In order to solve the original optimization problem and handle nonlinearities of the problem and multi-modalities of the fitness landscape, a sophisticated global optimization algorithm is required. In [7], a Genetic Algorithm has been proposed for this problem. In [8], a parallel genetic algorithm which employs a hybrid fine and coarse scale scheme has been presented. In [9] and [10], two cascaded approaches have been proposed: in the first two GAs are applied to the EIT problem to compute resistivity ranges and subsequently the resistivity distribution, in the second a Differential Evolution (DE) is employed to detect resistivity values in a known structure within a head model and the Newton–Raphson method is then activated for fine tuning of results found by the DE.

Taking into consideration the difficulties of this inverse problem, this paper proposes and compares two Memetic approaches, see [11]. Memetic Algorithms (MAs) are population based metaheuristics which contain one or more local search components integrated within the generation loop of an evolutionary framework. An important classification of MA structures is given in [12]: the local search can either be implemented within the variation operators (e.g. crossover or mutation) for controlling the offspring generation or as life-time learning i.e. on solutions already generated in order to improve upon their fitness values. Both the MAs proposed in this paper employ a DE based evolutionary framework, however the first MA integrates the local search within the variation operator, while the second applies a life-time learning to the generated solutions.

2 Problem Description

Fig. 1 shows an example of EIT setup. The object under study is circular and surrounded by $L = 16$ electrodes, attached to the surface of the object. One pair of electrodes is used for injecting a small current I into the object while the other pairs are used to measure voltage differences V between the electrodes. Since one of the pairs is linearly dependent, $L - 1 = 15$ different current patterns can be used. In this paper, we restrict our study to a situation where there is only one region of inhomogeneity which is assumed to be circular as well (two circles, one inside another). In addition, the resistivity values within and around the region are constant. Thus, the problem consists of detecting the circular area with a different and unknown resistivity value (e.g. the dotted circle in Fig. 1) whose location and size are also unknown, contained in a surrounding object characterized by an unknown resistivity value. Let $\Omega \in \mathbb{R}^2$ denote the internal region of the object under study whose boundary is denoted with $\partial\Omega$. The EIT problem can then be described by the so called complete electrode model (which is the most accurate EIT model description), see [13] and [14]:

$$\begin{aligned} \nabla \cdot (\sigma \nabla u) &= 0 \quad \text{in } \Omega \\ u + z_\ell \sigma \frac{\partial u}{\partial \mathbf{n}} &= U_\ell \quad \text{on } e_\ell, \quad \ell = 1, 2, \dots, L \\ \int_{e_\ell} \sigma \frac{\partial u}{\partial \mathbf{n}} dS &= I_\ell, \quad \ell = 1, 2, \dots, L \\ \sigma \frac{\partial u}{\partial \mathbf{n}} &= 0 \quad \text{on } \partial\Omega \setminus \bigcup_{\ell=1}^L e_\ell \end{aligned} \quad (1)$$

where $u = u(x_1, x_2)$ is the electric potential (voltage), $\sigma = \sigma(x_1, x_2)$ is a real-valued conductivity distribution (reciprocal to resistivity ρ , $\sigma^{-1} = \rho$), e_ℓ is the ℓ th electrode, z_ℓ is the real-valued effective contact impedance between e_ℓ and $\partial\Omega$, U_ℓ is the electrical potential on e_ℓ , I_ℓ is the injected current, and \mathbf{n} is the outward-pointing unit normal vector. Details on the derivative calculation in equation (1) are given in [14]. It must be observed that the conservation of charge dictates that $\sum_{\ell=1}^L I_\ell = 0$. In addition, the reference point, for potentials on the electrode ground voltage, must be selected. In this paper, without a generality loss, the sum of potentials is set to zero: $\sum_{\ell=1}^L U_\ell = 0$.

The forward problem has been artificially generated by constructing a mesh in the external object and pseudo-randomly locating a structure within it. Resistivity values of both the objects are also pseudo-randomly generated. After the

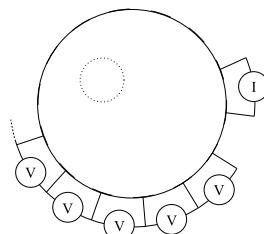


Fig. 1. Electrical impedance tomography setup

problem generation, the first mesh is ignored and for the inverse problem a new mesh is built up. The use of different meshes for forward and inverse problems has been done in order to make our application more realistic, since in the real-world the object would be continuous (this explains a finer mesh in the forward problem) and only a discrete approximation would be available to describe its features (see "inverse crime" [15]). Fig. 2 shows both meshes employed in the forward and inverse problems.

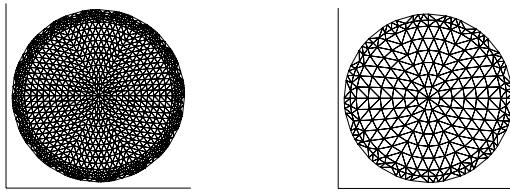


Fig. 2. Meshes used in forward (left) and inverse (right) calculations

Let us call the internal circular area that possesses a different resistivity from the background, inhomogeneous region. Let us define r_* and r_Ω the radii of the inhomogeneous area and the external object respectively; (α, d) the angle and radius of the polar coordinates (the origin of the polar system is located in the center of the external object and the reference axis is horizontal) within the center of the inhomogeneous region, respectively; ρ_* and ρ_Ω , the resistivity values of the inhomogeneous and surrounding regions, respectively. A structural configuration and thus a candidate solution of our problem is identified by a vector $\mathbf{x} = \langle \alpha, d, r_*, \rho_*, \rho_\Omega \rangle$ and the corresponding decision space is given by: $D = [0, 2\pi] \times [0, r_\Omega] \times [0, r_\Omega] \times [100, 1000] \times [100, 1000]$. The lower and upper bounds 100 and 1000 define the range of resistivity values of interest for this application. By means of the candidate solution x , a discrete approximation σ^h of the conductivity distribution σ is then computed for the nodes of the inverse problem mesh. Then, the EIT inverse problem consists in minimizing the following objective function:

$$F(\sigma^h) = \sum_{k=1}^{L-1} \|V^k - U(\sigma^h)^k\|^2 = \sum_{k=1}^{L-1} \sum_{\ell=1}^L (V_\ell^k - U(\sigma^h)_\ell^k)^2 \quad (2)$$

where k denotes the current contribution due to each pair of electrodes, V_ℓ^k is the measured voltage at the ℓ th electrode with the k th current contribution and $U(\sigma^h)_\ell^k$ the corresponding computed voltage.

3 The Proposed Memetic Algorithms

In order to minimize the fitness function in (2), two MAs have been implemented. Both the algorithms employ a self-adaptive differential evolution scheme, in the

same fashion as the algorithm proposed in [16], as an evolutionary framework. The first MA, namely Variation Operator Local Search Differential Evolution (VOLSDE) performs a local search on the scale factor during generation of the offspring. The second MA, namely Life-time Learning Local Search Differential Evolution (LLSDE) performs a local search on the individual already generated.

3.1 Evolutionary Framework

An initial population of S_{pop} individuals is pseudo-randomly generated. Each individual (the i^{th} in the population) is composed, in a self-adaptive logic, of its genotype and its control parameters $x_i = \langle \alpha_i, d_i, r_{*i}, \rho_{*i}, \rho_{\Omega i}, F_i, CR_i \rangle$, where each design variable is sampled within its corresponding decision interval (e.g. α_i is sampled within $[0, 2\pi]$) and F_i and CR_i are sampled between 0 and 1. In accordance with a self-adaptive logic the variation operations are preceded by the parameter update. More specifically when, at each generation, the i^{th} individual x_i is taken into account and three other individuals are extracted pseudo-randomly, its parameters F_i and CR_i are updated according to the following scheme:

$$F_i = \begin{cases} F_l + F_u rand_1, & \text{if } rand_2 < \tau_1 \\ F_i, & \text{otherwise} \end{cases} \quad (3)$$

$$CR_i = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_i, & \text{otherwise} \end{cases} \quad (4)$$

where $rand_j$, $j \in \{1, 2, 3, 4\}$, are uniform pseudo-random values between 0 and 1; τ_1 and τ_2 are constant values which represent the probabilities that parameters are updated, $F_l = 0.1$ and $F_u = 0.9$ are constant values which represent the minimum value that F_i could take and the maximum variable contribution to F_i , respectively. The newly calculated values of F_i and CR_i are then used for generating the offspring. For each individual x_i of the S_{pop} available, three individuals x_r , x_s and x_t are pseudo-randomly extracted from the population. A provisional offspring x'_{off} is then generated by mutation as $x'_{off} = x_t + F_i(x_r - x_s)$. When the mutation occurs, to increase exploration, each gene of the new individual x'_{off} is switched with the corresponding gene of x_i with a uniform probability CR_i and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{i,j} & \text{if } rand(0, 1) < CR_i \\ x'_{off,j} & \text{otherwise} \end{cases} \quad (5)$$

where $rand(0, 1)$ is a random number between 0 and 1; j is the index of the gene under examination. The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning logic, it replaces x_i if and only if $f(x_{off}) < f(x_i)$; otherwise no replacement occurs.

3.2 Variation Operator Local Search

The Variation Operator Local Search Differential Evolution (VOLSDE) corrects the evolutionary framework in Subsection 3.1 by including a local search application as a possibility for choosing F_i . The main idea is that updating of the

scale factor and thus generation of the offspring is, with a certain probability, controlled in order to guarantee a high quality solution which can have a key role in subsequent generations, see also [12].

For given values of x_r , x_s , x_t , and the design variables undergoing switching due to crossover, the generation of a new offspring is seen as a procedure depending on the scale factor F_i . We propose applying a local search to F_i in order to detect the scale factor value which guarantees an offspring with high performance. In other words, we aim at minimizing the function $f(F_i)$ in the decision space $[-1, 1]$. A negative scale factor means that the search direction is inverted. If a search in the negative direction succeeds, the corresponding positive value is assigned to the offspring for subsequent generations. For sake of clarity, the procedure describing the fitness function of the variation operator local search is given in the following pseudocode.

```

insert  $F_i$ ;
compute  $x'_{off} = x_t + F_i(x_r - x_s)$ ;
perform the swapping of the genes
and generate in a crossover fashion  $x_{off}$ ;
compute  $f(F_i) = f(x_{off})$ ;
return  $f(F_i)$ ;

```

In order to perform this minimization, the Sequential Quadratic Programming proposed in [17] has been applied. This local search is applied for 20 fitness evaluations with a uniform probability equalling 0.1 to the scale factor of the best performing individual of the population.

3.3 Life-Time Learning

The Life-time Learning Local Search Differential Evolution (LLSDE) attempts to improve upon the algorithm described in Subsection 3.1 by applying a local search on an individual of the population. At each generation, with a uniform probability equal to 0.2, one individual is pseudo-randomly selected and undergoes the Nelder Mead Algorithm (NMA) [18] for 50 fitness evaluations. Thus, on average, one NMA local search is performed every five generations. The individual that has undergone this local search is then inserted into the population and the original genotype updated.

3.4 Variation Operator Local Search vs. Life-Time Learning

Although the VOLSDE and LLSDE employ the same evolutionary framework and both attempt to enhance its performance by means of a local search, they have important conceptual and computational differences. The VOLSDE takes into consideration that, as shown in [16], the DE might be subject to stagnation problems and that a periodic update of control parameters is beneficial in terms of successful generation of the offspring. On the other hand, the update scheme proposed in [16] can turn out to be too similar to a randomized search and a periodic application of a local search can lead to the generation of an offspring solution with high performance. It must be observed that the local search in this

algorithm is applied to the scale factor and can thus be seen as a component integrated into the DE logic which empowers the DE search. On the contrary, the LLLSDE relies on the fact that an alternative search logic supports and compensates the DE logic with the aim of exploring some regions of the decision space which would otherwise be neglected. In this light the choice of applying the variation operator local search to the best individual can be seen as an attempt to obtain maximum profit from the DE logic, while application of the NMA to a pseudo-randomly selected individual relies on the philosophy that each solution can potentially lead to a significant improvement if an alternative exploratory logic is tested.

As regards computational cost, the VOLSDE performs the search on a univariate function regardless of the amount of decision variables, while the LLLSDE performs a local search along all variables of the decision space. This fact means that the LLLSDE employs a local search that is clearly more expensive than the VOLSDE and, in other words, needs to devote a relatively high computational budget to the local search in order to obtain some improvements. In addition, since the VOLSDE employs for global and local search similar search logics, there is no need for a frequent local search application (the 0.1 probability setting). This fact can be justified with the consideration that if a local search is successful and an offspring solution with high performance is generated, some generations are required in order to allow proper propagation of the promising genotype. Regarding the LLLSDE, since the NMA offers a different search perspective compared to the DE, a relatively frequent (probability 0.2) local search application is required in order to maintain a competitive/cooperative search balance between global and local components, see [19] and [20].

4 Numerical Results

Five internal object configurations have been pseudo-randomly generated. The VOLSDE and the LLLSDE have been tested for these five cases. The performance of the proposed algorithms has been compared with the three following metaheuristics: 1) a real-valued Genetic Algorithm (GA) employing the blend crossover proposed in [21] with probability 0.8 and a uniform mutation with probability 0.1; 2) a plain Differential Evolution (DE) with $F = 0.8$ and $CR = 0.5$ in accordance to the suggestions given in [22]; 3) the Self-Adapting Control Parameters Differential Evolution (SACPDE) proposed in [16]. For each algorithm, a population size equal to 20 has been set. Each algorithm has been run 25 times with a budget condition of 5000 fitness evaluations.

Table 1 lists the best solutions detected by each algorithm, their corresponding fitness values f , and the average fitness values $\bar{f} \pm$ the standard deviation error σ (over the 25 independent runs) for the five test cases analyzed. In addition, in order to execute a numerical comparison of the convergence speed performance, the Q measure (Q stands for Quality) has been computed. The Q measure is defined as $Q = \frac{\bar{ne}}{R}$ where \bar{ne} is the average amount of fitness evaluations required to reach a predefined threshold value THR and the robustness R is the percentage of runs when this threshold is reached. In this application, THR has been

Table 1. Numerical results for the five test cases

	α	d	r_*	ρ_*	ρ_Ω	f	$f \pm \sigma$	Q
GA	5.520	12.900	2.169	468.831	722.854	3.757E04	4.030E04 \pm 2.539E04	900.500
DE	5.520	13.783	2.420	425.534	724.710	1.881E04	7.186E03 \pm 2.652E04	29.079
SACPDE	5.523	13.840	2.472	425.541	724.710	1.881E04	2.151E03 \pm 1.200E03	18.719
VOLSDE	5.522	13.938	2.258	425.542	724.710	1.881E04	1.887E03 \pm 2.129E01	19.327
LLSDE	5.525	13.518	2.180	425.536	724.710	1.881E04	1.882E03 \pm 1.753E-02	19.665
GA	6.224	6.964	7.954	276.093	530.917	2.601E03	4.624E04 \pm 4.490E04	inf
DE	6.232	7.327	7.684	263.469	529.645	1.781E02	5.082E03 \pm 2.218E04	33.373
SACPDE	6.231	7.342	7.675	263.683	529.592	1.768E02	2.110E03 \pm 3.119E03	24.432
VOLSDE	6.234	7.343	7.689	263.682	529.592	1.768E02	2.327E03 \pm 3.118E03	28.989
LLSDE	6.231	7.346	7.673	263.682	529.591	1.768E02	1.365E03 \pm 2.409E03	23.777
GA	3.673	3.350	10.231	318.511	618.155	2.935E03	6.571E04 \pm 5.743E04	731.750
DE	3.687	3.439	10.040	310.101	614.232	1.838E03	1.794E04 \pm 8.021E04	31.452
SACPDE	3.679	3.425	10.027	309.928	614.185	1.836E03	1.981E03 \pm 3.812E02	18.507
VOLSDE	3.684	3.423	10.030	309.928	614.186	1.836E03	1.921E03 \pm 2.816E02	19.116
LLSDE	3.670	3.420	10.024	309.929	614.185	1.836E03	1.899E03 \pm 2.953E02	19.412
GA	4.153	10.778	12.968	402.021	801.017	2.191E03	1.645E05 \pm 3.799E05	1219
DE	4.154	10.384	12.771	402.324	806.481	9.702E02	1.301E05 \pm 6.454E05	22.338
SACPDE	4.157	10.414	12.820	402.328	806.496	9.701E02	1.050E03 \pm 2.111E02	14.574
VOLSDE	4.158	10.385	12.783	402.327	806.496	9.701E02	1.033E03 \pm 2.289E02	15.410
LLSDE	4.157	10.384	12.801	402.327	806.496	9.701E02	1.060E03 \pm 3.508E02	14.617
GA	3.070	2.425	6.288	234.968	287.279	8.290E01	4.989E03 \pm 6.757E03	18.163
DE	3.088	2.569	3.209	132.980	287.271	2.005E00	1.976E01 \pm 6.235E01	11.828
SACPDE	3.069	2.486	3.175	131.833	287.324	1.374E00	1.154E01 \pm 7.662E00	6.085
VOLSDE	3.081	2.404	3.187	131.833	287.324	1.374E00	3.261E01 \pm 1.112E02	6.690
LLSDE	3.046	2.490	3.194	131.836	287.323	1.374E00	1.349E01 \pm 7.280E00	7.854

set, for each test case, equal to the best, overall, reached fitness value +2000. The value inf means that $R = 0$, i.e. the algorithm never reached the THR . The best results are highlighted in bold face.

Numerical results show that the DE approach leads to a significantly better performance than a real-valued GA and the self-adaptation seems beneficial for the class of problems under investigation. On the contrary, the role of the local search seems to be rather controversial since its application for some test cases (e.g. test case 2) leads to an improvement in the algorithmic performance whilst for one test case (i.e. test case 5) it leads to a worsening of the algorithmic performance. According to our interpretation, with reference to Table 1, the original positioning and size of the object with inhomogeneous resistivity heavily affects features of the fitness landscape and thus the algorithmic performance of the various algorithms. More specifically, if the inhomogeneous region is large and located close to the center of the external object, the optimization problem is not very difficult and a global optimizer without local search turns in a good performance. On the other hand, if the inhomogeneous region is small and located far from the center, the optimization problem is more challenging and a Memetic approach seems to be more efficient. Finally, the comparison of VOLSDE and LLSDE confirms that the VOLSDE, as an empowerment of the SACPDE, improves upon the SACPDE performance for inhomogeneous small areas located in a peripheral region of the external object. The LLSDE seems to have a rather robust behavior and converged on solutions with the best performance in three test cases out of the five analyzed. An example of best solution and average performance trend are graphically represented in Fig. 3 and Fig. 4, respectively.

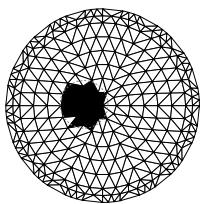


Fig. 3. Example of best solution

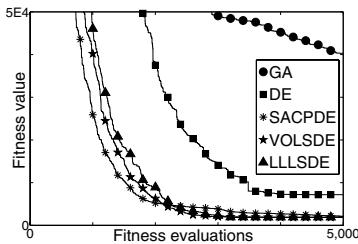


Fig. 4. Example of Performance Trend

5 Conclusion

This paper proposes two MA implementations for solving EIT inverse problems. The first implementation integrates the local search within the generation operator of a self adaptive DE proposed in literature, the second attempts to improve upon the solution by applying a local search to an already generated individual in the spirit of life-time learning. Numerical results show that the fitness landscape is influenced by the location and size of the inhomogeneous area. The proposed algorithms have good performance, especially in the more hard to solve cases, when the inhomogeneous area is small and located far from the center of the external object. The MA with life-time learning local search seems to have, on average, the best performance. On the other hand, a complex geometry would lead to a high dimensionality, which would increase the computational cost of the life-time learning local search and likely make more appealing the employment of a variation operator local search.

References

1. Henderson, R., Webster, J.: An impedance camera for spatially specific measurements of the thorax. *IEEE Trans. of Biom. Eng.* 25, 250–254 (1978)
2. Vauhkonen, M., Vadász, D., Karjalainen, P.A., Somersalo, E., Kaipio, J.P.: Tikhonov regularization and prior information in electrical impedance tomography. *IEEE Trans. on Medical Imaging* 17, 285–293 (1998)
3. Neittaanmäki, P., Rudnicki, M., Savini, A.: *Inverse Problems and Optimal Design in Electricity and Magnetism*. Oxford University Press, Oxford (1996)
4. Calderón, A.P.: On an inverse boundary value problem. *Computational and Applied Mathematics* 25(2–3), 133–138 (2006)
5. Uhlmann, G.: Developments in inverse problems since Calderón’s foundational paper. In: Christ, M.E., Kenig, C.E. (eds.) *Harmonic Analysis and Partial Differential Equations*, ch. 19. University of Chicago (1999)
6. Cheney, M., Isaacson, D., Newell, J., Simske, S., Goble, J.: NOSER: An algorithm for solving the inverse conductivity problem. *IJIST* 2, 66–75 (2005)
7. Cheng, K.-S., Chen, B.-H., Tong, H.-S.: Electrical impedance image reconstruction using the genetic algorithm. In: Proc. of 18th Ann. Int. Conf. IEEE Eng. in Med. and Bio. Soc., pp. 768–769 (1996)

8. Carosio, G., Rolnik, V., Seleg him, P.J.: Improving efficiency in electrical impedance tomography problem by hybrid parallel genetic algorithm and a priori information. *Journal of Computational Physics* 173, 433–454 (2001)
9. Olmi, R., Bini, M.: A genetic algorithm approach to image reconstruction in electrical impedance tomography. *IEEE Trans. EC* 4(1), 83–88 (2000)
10. Li, Y., Rao, L., He, R., Xu, G., Wu, Q., Yan, W., Dong, G., Yang, Q.: A novel combination method of electrical impedance tomography inverse problem for brain imaging. *IEEE Trans. Magnetics* 41, 1848–1851 (2005)
11. Kononova, A.V., Ingham, D.B., Pourkashanian, M.: Simple scheduled memetic algorithm for inverse problems in higher dimensions: Application to chemical kinetics. In: Proc. of the IEEE WCCI, pp. 3906–3913 (2008)
12. Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing. *ECJ* 12(3), 273–302 (2004)
13. Vauhkonen, M., Kaipio, J.P., Somersalo, E., Karjalainen, P.A.: Electrical impedance tomography with basis constrains. *Inv. Pr.* 13, 523–530 (1997)
14. Cheney, M., Isaacson, D., Newell, J.: Electrical impedance tomography. *SIAM review* 14, 85–101 (1999)
15. Kaipio, J., Somersalo, E.: Statistical inverse problems: Discretization, model reduction and inverse crimes. *JCAM* 198(2), 493–504 (2007)
16. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. EC* 10(6), 646–657 (2006)
17. Biggs, M.: Constrained minimization using recursive quadratic programming. In: Dixon, L., Szergo, G. (eds.) *Towards Global Optimization*, pp. 341–349 (1975)
18. Nelder, J., Mead, R.: A simplex method for function optimization. *Computation Journal* 7, 308–313 (1965)
19. Ong, Y.S., Keane, A.J.: Meta-lamarkian learning in memetic algorithms. *IEEE Trans. on EC* 8(2), 99–110 (2004)
20. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. *IEEE Trans. on SMC-B* 37(1), 28–41 (2007)
21. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In: *Foundations of GAs* 2, pp. 187–202. Morgan Kaufmann, San Francisco (1993)
22. Zielinski, K., Weitkemper, P., Laur, R., Kammeyer, K.-D.: Parameter study for differential evolution using a power allocation problem including interference cancellation. In: Proc. of the IEEE CEC, pp. 1857–1864 (2006)

Estimating HMM Parameters Using Particle Swarm Optimisation

Somnuk Phon-Amnuaisuk

Perceptions and Simulation of Intelligent Behaviours,
Multimedia University, 63100 Cyberjaya, Selangor Darul Ehsan, Malaysia
somnuk.amnuaisuk@mmu.edu.my

Abstract. A Hidden Markov Model (HMM) is a powerful model in describing temporal sequences. The HMM parameters are usually estimated using Baum-Welch algorithm. However, it is well known that the Baum-Welch algorithm tends to arrive at local optimal points. In this report, we investigate the potential of the Particle Swarm Optimisation (PSO) as an alternative method for HMM parameters estimation. The domain in this study is the recognition of handwritten music notations. Three observables: (i) sequence of ink patterns, (ii) stroke information and (iii) spatial information associated with eight musical symbols were recorded. Sixteen HMM models were built from the data. Eight HMM models for eight musical symbols were built from the parameters estimated using the Baum-Welch algorithm and the other eight models were built from the parameters estimated using PSO. The experiment shows that the performances of HMM models, using parameters estimated from PSO and Baum-Welch approach, are comparable. We suggest that PSO or a combination of PSO and Baum-Welch algorithm could be alternative approaches for the HMM parameters estimation.

1 Background

This paper discusses our application of PSO [8] in estimating transition matrix A and emission matrix B of *Hidden Markov Models (HMMs)*. In PSO approach, each particle in the swarm searches for solutions in the search space with the help of the information coming from (i) the best experience of each particle and (ii) the best experience learned from its neighbor (within their observation range). This kind of heuristic has an advantage of being generic, hence, if domain specificity is not required this class of problem solver is attractive. There are many reports claiming that PSOs have performed very well in many domains. This study aims to investigate the application of PSO to improve the estimation of HMM parameters.

To date, there is no commercial package that allows users to enter music notations naturally as handwritten musical symbols. Available systems either support the entering of notations via predefined symbol pallets, or predefined handwritten gestures which users need to learn to associate the gestures with the intended musical symbols. There are two main approaches in the recognition of

handwritten music notations task. One is the *Optical Music Recognition (OMR)* which recognises printed music [17] and the other is the pen-based approach [16] which usually utilises temporal information obtained during the writing process. Researchers commonly resort to the HMM approach in modelling temporal sequences obtained from handwritten symbols. Baum-Welch algorithm is often the researchers' choice for the parameters estimation of HMMs. Here, we investigate the application of PSO as an alternative approach.

We organise the presentation into the following topics: 1. Background, 2. Literature review, 3. Estimating HMM parameters using PSO, 4. Results and 5. Conclusion.

2 Literature Review

Particle swarm optimization (PSO) [8] is one of the evolutionary computation techniques inspired from social behaviours observed in animals such as bird flocking, fish schooling, etc. There is no central control in this computation paradigm. The overall behaviour emerges as a collective behaviour of the whole population. In PSO, interactions among particles (members of the population) are governed by the so called cognitive learning part and social learning part of each particles. The cognitive learning part relates to a particle's own experience and the social learning part relates to positive experiences of others. The idea behind PSO is simple and yet PSO has been applied to many problem domains successfully, such as in image processing, data clustering, DNA sequence alignment and optimisation of ANN, SVM weights [20]. In the recent work by [21], PSO has been used to estimate optimal parameters of HMM in speech recognition tasks. In this work, we investigate the optimisation of HMM parameters in handwritten recognition tasks.

The work in handwritten recognition can be traced back to the 1960s [19]. The area was inactive for a while during the 1980s. However, in the past decade, the topic has regained interest from researchers [3], [12], [13]. This could be mainly due to the advances in computing power, advances in pen-input devices and better understanding of machine learning techniques.

To date, specific applications in signature verification and postcode reading have been successfully implemented. However, to my knowledge, the application of handwritten music recognition is still being researched [1], [2], [4], [5], [9], [11], [14]. From the literature, there are two main categories of works in this area, recognising image of musical symbols [17] and recognising digital ink of handwritten music notation [13].

From the literature, ANN and HMM are the main techniques [10], [16], [15] in the majority of works in handwritten music recognition. HMM is a popular technique in modelling temporal sequence since temporal information in the observed sequence could be captured as transition probabilities in the hidden nodes. The conditional observation probability gives more expressiveness than a simple Markov Model. Although HMM is a very popular technique in speech recognition, handwritten recognition, cryptanalysis, machine translation, etc.,

there is no known method for designing an optimal HMM structure and its parameters. This is still an open area and researchers have tried many tactics apart from the well known Baum-Welch algorithm, such as the Bayesian approach and evolutionary approach [18].

3 Estimating HMM Parameters Using PSO

3.1 Problem Domain and HMM

In our setup, handwritten music notations were entered naturally using a pen-tablet. Nine digital ink primitives were recorded (i.e., eight directions: east, southeast, south, southwest, west, etc., and a dot). A sequence of observable symbols for a music symbol was grouped and preprocessed into a canonical form (i.e., standard representation for a particular symbol). As a result, the sequence of observable symbol was invariant from the order of input strokes in different writing styles.

For each music notation, we built a HMM model where we assumed that the observable symbols were dependent on a set of n hidden nodes. We gave the formal definition of the HMM as below.

Definition 1. HMM: Let HMM be defined using triple (Π, A, B) , where Π is the initial state probabilities, A is the state transition matrix and B is the emission matrix. Matrix A describes the transitional behavior of the hidden state and Matrix B describes the coupling between the hidden state and the observable states.

In our formulation, see Figure 1(a), the transition matrix ($a_{ij} \in A$) of the hidden states captured the temporal characteristic (i.e. transition probability) of each model θ_h e.g., $p(\omega_{t+1}|\omega_t, \theta_h)$. The emission matrix ($b_{jk} \in B$) captured the likelihood of observable symbols (i.e., ink directions, spatial information and stroke information) in each hidden state $p(v_t|\omega_t, \theta_h)$. Hence the observed sequence V^T altogether with the hidden sequence Ω^T might be used to predict the likelihood of a model.

The value a_{ij} of the transition matrix A and the values b_{jk} of the observation matrix B may be learned from the training data (i.e., observed ink directions)

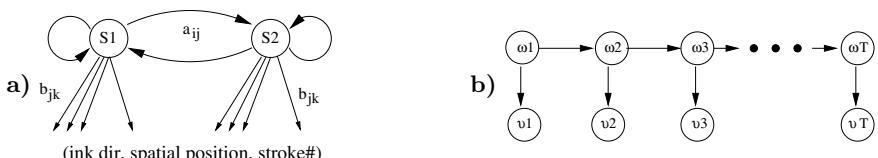


Fig. 1. (a) HMM with two hidden states; (b) $p(V^T|\theta_h)$ is estimated from a sequence of observations V^T coupled with the hidden states for a given model θ_h

using *Baum-Welch* (BW) algorithm (follow [7]). The values of γ_{ij} (i.e., an estimated probability that the HMM was in the state ω_i at time $t - 1$ and in the state ω_j at time t) were estimated from a given training sequence. The $\alpha_i(t - 1)$ and $\beta_j(t)$ were forward and backward estimations that the HMM was in the hidden state $\omega_i(t - 1)$ and $\omega_j(t)$ respectively.

$$\gamma_{ij} = \frac{\alpha_i(t - 1)a_{ij}b_{jk}\beta_j(t)}{P(V^T|\theta_h)} \quad (1)$$

$$a_{ij} = \frac{\sum_{t=1}^T \gamma_{ij}(t)}{\sum_{t=1}^T \sum_m \gamma_{im}(t)}; \quad b_{jk} = \frac{\sum_{t=1}^T b_{jk}(t)}{\sum_{t=1}^T \sum_m b_{jm}(t)} \quad (2)$$

However, there is no known technique for obtaining the optimal values of A and B matrices. The most popular technique for estimating these parameters is the Baum-Welch algorithms. Unfortunately, it is well known that BW tends to get trapped in the local optima.

3.2 Particles Swarm Optimisation

Definition 2. Particle Swarm Optimisation (PSO): Let each PSO particle be defined using quadruple $x_{ij}(t), v_{ij}(t), p_{ij}(t), g_{ij}(t)$. The subscript i, j denotes the particle i and the dimension j . The interpretation of these parameters are as follows: $x_{ij}(t)$ is the current position of the particle i in the dimension j ; $v_{ij}(t)$ is the current velocity; $p_{ij}(t)$ is the cognitive learning part which is interpreted as the previous best position and $g_{ij}(t)$ is the social learning part which is interpreted as the global best position.

Particles fly through the search space using the following update rules:

$$x_{ij}(t + 1) = x_{ij}(t) + v_{ij}(t + 1) \quad (3)$$

$$v_{ij}(t + 1) = wv_{ij}(t) + c_1r_{1j}(p_{ij}(t) - x_{ij}(t)) + c_2r_{2j}(g_{ij}(t) - x_{ij}(t)) \quad (4)$$

where w is the inertia weight, v_{ij} is the velocity of a particle i in a j dimension. c_1, c_2, r_{1j}, r_{2j} are the acceleration coefficients (note that random coefficient r_{1j} and r_{2j} explicitly promote randomness in each dimension). Exploitation and exploration are controlled through the cognitive and social components. There are two common information sharing among particles, the star topology and the ring topology neighborhood. Here, we decided to use a star topology neighborhood where information regarding the global best position is communicated to all the particles in the swarm.

Representation of PSO: In this implementation, each particle i was a vector of j dimension. Each $x_{ij} \in \Re$ represented parameters Π, A, B of a HMM. For example, a HMM with two hidden states and five observable variables would be represented as a string X_i where $|X_i| = 16$ (i.e., $2 + 4 + 10; |\Pi| + |A| + |B|$).

Coding and Decoding HMM parameters: Each x_{ij} could have any values in \mathfrak{R} . However, only positive $|x_{ij}|$ values would be used when decoding from PSO to HMM. Normalisation would ensure that the decoded HMM has well-formed probability entries. The encoding of PSO was by merely flattening the HMM parameter matrices into a string.

Updating P_i and G_i : Each particle in the swarm represented one plausible set of HMM parameters. At every iteration, each particle was attracted by its best experience and the best experience of the swarm. The fitness of a particle was measured using:

$$\text{fitness}(X_i) = \frac{1}{n} \sum_{s=1}^n p(V^T | \theta_h) \quad (5)$$

where n was the number of observed sequences used to determine the fitness and θ_h was the HMM model built from the particle X_i . The best experience of the particle P_i and the best experience of the swarm G_i were updated using:

$$P_i = \max_i(\text{fitness}(X_i(t)), \text{fitness}(X_i(t-1))) \quad (6)$$

$$G_i = \text{argmax}_i \text{fitness}(X_i(t)) \quad (7)$$

3.3 Experimental Setup

Eight musical symbols (i.e., from left to right and top to bottom: semibreve, minim, crotchet, quaver, crotchet-rest, flat, natural and sharp, see Figure 2) were used in this experiment. Fifty samples for each musical symbol were collected from a single user. Three observed variables were utilised in the recognition process. They were (i) ink directions, (ii) spatial information and (iii) stroke information (i.e., the numbers of pen-down, pen-up pairs).

In each training and testing cycle, 25 training samples and 25 testing samples were randomly selected (sampling without replacement). The experiment was



Fig. 2. Music symbols used in this experiment

repeated for 20 times. So for each musical symbol, there were a total of twenty sets of 25 training patterns and twenty sets of 25 testing patterns.

For each training and testing run (20 runs in total), sixteen HMM models were built. Eight HMM models were built for each musical symbols using the parameters estimated from the standard Baum-Welch algorithm. In our implementation, in each iteration, each training sample was fed to the BW for three consecutive times. If the BW did not converge after being trained with all samples, the process would be repeated. The maximum iteration for BW was set at 100 in this experiment.

The other eight HMM models were built using parameters obtained from PSO. PSO parameters were determined empirically. However, there were many possible combinations and there was no known method for setting up these parameters. In this experiment the following parameter settings were used (see Table 1).

Table 1. Parameters setting in this study

Parameters	Values
HMM : Hidden states	2
Observed variable one (ink directions)	9
Observed variable two (spatial positions)	6
Observed variable three (pen strokes)	3
PSO : Swarm size	40
Inertia weight w	[0.5 1]
Acceleration c_1	2
Acceleration c_2	(0, 0.5, 1.0, 1.5, 2.0, 3.0)
Maximum PSO iteration (for each run)	50
Total runs	20

In this experiment, we deliberately fixed the number of hidden states in the HMM to two states. We expected to see the effects of PSO more clearly by minimising the number of hidden states.

4 Results and Discussion

The performance of the HMMs learned by Baum-Welch and by PSO were compared using *sensitivity* and *specificity* measurements.

$$Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

4.1 Observing the Effects of Exploitation and Exploration of the Swarm

In this empirical study, we started based on the intuition that different exploitation and exploration of the search space might contribute to the quality of the solution found. In PSO, the exploitation and exploration rate of the swarm could be controlled by varying the values of c_1 and c_2 respectively (see equation 4). Setting the value of c_1 to 0 meant no exploitation and setting the value of c_2 to 0 meant no exploration while increasing this value increased the exploitation and exploration of the swarm. Here, we fixed c_1 to 2.0¹ and varied c_2 from 0, 0.5, 1.0, 1.5, 2.0 and 3.0. The sensitivity of PSO under a fixed c_1 and different c_2 are shown in Figure 3. From our observation, there was no strong impression about the effects of c_1 and c_2 . We also found that the hill climbing rate of PSO died out fast (i.e., the updating of a better position decreased fast). We believe that this was due to a large search space since possible values of a_{ij} and b_{jk} were in \mathbb{R} . This was hard for the swarm and extra knowledge may be needed to guide the search. However, when comparing the HMM models built from parameters estimated using BW and PSO, we found that, there was some improvement on the PSO version as compared to the BW version (see Figure 4 and Table 2).

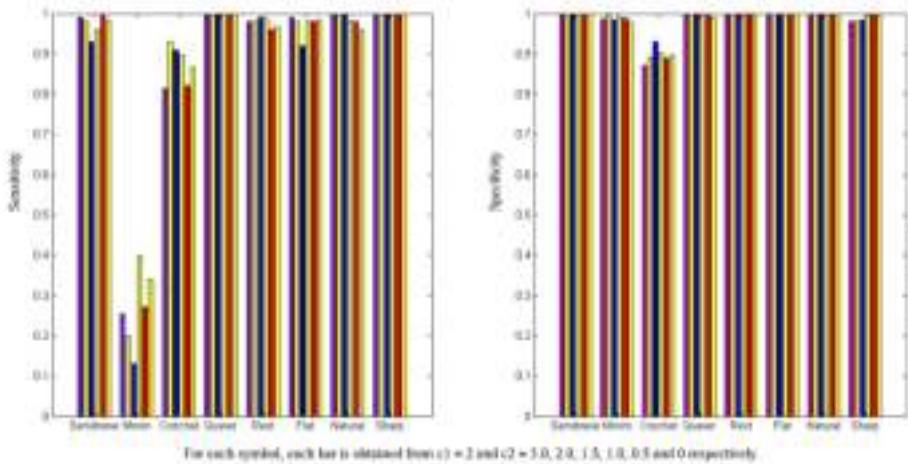


Fig. 3. Sensitivity from the HMM models built from parameters estimated using PSO. Note that in each subgroup (each symbol), each bar presented the sensitivities obtained from a fixed c_1 and varied c_2 parameters of the PSO.

From Figure 4, there is no clear distinction in the specificity values between PSO and BW. However, PSO seems to be slightly more sensitive than BW for some symbols. In order to justify the statistical significance of the sensitivity

¹ Here, we fixed the exploitation and varied the exploration of the PSO.

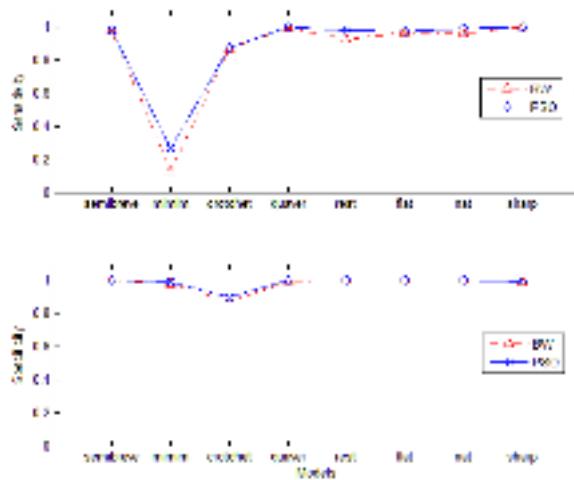


Fig. 4. Average sensitivities and specificities from HMM models built from parameters estimated using BW and PSO

Table 2. Results from a one tailed t -test of the HMM sensitivity

Symbols	Sensitivity		Variance		df = 5	t -value	p -value
	μ_{BW}	μ_{PSO}	BW	PSO			
Semibreve	0.975	0.974	0.0002	0.0006	0.423	0.34	
Minim	0.129	0.265	0.0008	0.0092	-3.178	0.01	
Crotchet	0.868	0.873	0.0012	0.0023	-0.786	0.23	
Quaver	0.995	1.000	0.0000	0.0000	-1.463	0.10	
Rest	0.925	0.977	0.0036	0.0001	-1.885	0.05	
Flat	0.968	0.972	0.0033	0.0007	-0.304	0.38	
Natural	0.962	0.987	0.0006	0.0003	-0.994	0.18	
Sharp	1.000	1.000	0.0000	0.0000	0.000	0.50	

obtained from the PSO version, a one-tailed t -test was performed. The null hypothesis stated that $H_0 : \mu_{pso} = \mu_{bw}$ and the alternative hypothesis stated that $H_a : \mu_{pso} \neq \mu_{bw}$. The result (see Table 2) showed that at the 95% confidence, we would accept the null hypothesis for all symbols except the minim and the crotchet-rest. Although not conclusive, it seemed PSO could improve HMM parameters on the recognition of the minim and the crotchet-rest while performing equally well for other symbols.

5 Conclusion and Further Work

PSO performs parallel search on the search space and the behaviour of its particles is governed by the weight w and acceleration parameter c_1, c_2 in equation 4,

PSO shared many similarity with Simulated Annealing (SA) except that, PSO performs a parallel search and the search direction is loosely governed by past experience from individual particle and from the swarm.

The fact that PSO does not have many control parameters makes the approach appealing but at the same time it also limits the expressiveness of the control to a certain degree. In PSO, a good combination of both exploitation and exploration are crucial. However, there are many plausible settings for these parameters and it is not conclusive what the appropriate values are. In this light, adaptive setting of exploitation or exploration level could be an interesting further work from this stage.

Another extension of this work would be to improve on the guiding knowledge. Incorporating BW during each PSO iteration could be one possible extension, instead of attracting particles with P_i and G_i alone. At each iteration, particles would be optimised using BW. BW acts as a local search in this sense and PSO serves mostly as a means to explore the whole search space.

In this report, we investigate the potential of PSO as an alternative approach to learn HMM parameters. We have shown that PSO can learn matrices Π , A and B . The obtained values are comparable or better than the values obtained using standard Baum-Welch algorithm.

Acknowledgement. I would like to thank the reviewers for their useful comments. I would also like to thank Lee Kian Chin for his help in preparing the data set.

References

1. Anstice, J., Bell, T., Cockburn, A., Setchell, M.: The design of a pen-based musical input system. In: Proceedings of the Sixth Australian Conference on Computer-Human Interaction, pp. 260–267. IEEE Computer Society, Los Alamitos (1996)
2. Bainbridge, D., Bell, T.: The challenge of optical music recognition. Computers and the Humanities 35, 95–121 (2001)
3. Blostein, D., Baird, H.S.: A critical survey of music image analysis. In: Baird, Bunke, Yamamoto (eds.) Structured Document Image Analysis. Springer, Heidelberg (1992)
4. Brown, A.R.: Composing by pen: Exploring the effectiveness of a system for writing music notation on computers via pen input. Australian Journal of Music Education, AMSE (1), 48–56 (1998)
5. Buxton, W., Sniderman, R., Reeves, W., Patel, S., Baecker, R.: The evolution of the SSSP score editing tools. Computer Music Journal, CMJ 3(4), 14–25 (1979)
6. Connell, S.D., Jain, A.K.: Writer adaptation for online handwriting recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence 24(3), 329–346
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, Chichester (2000)
8. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
9. Forsberg, A., Dieterich, M., Zeleznik, R.: The music notepad. In: Proceedings of the User Interface Software Technology, ACM SIGGRAPH, pp. 203–210 (1998)

10. George, S.E.: Online pen-based recognition of music notation with artificial neural networks. *Computer Music Journal*, CMJ 27(2), 70–79 (2003)
11. George, S.E.: Pen-based input for on-line handwritten music notation. In: George, S.E. (ed.) *Visual Perception of Music notation: On-line and off-line recognition*, Melbourne. IRM Press (2005)
12. Kam-Fai, C., Dit-Yan, Y.: Mathematical expression recognition: A survey. *International Journal on Document Analysis and Recognition*, IJDAR 3(1), 3–15 (2000)
13. Kavallieratou, E., Fakotakis, G., Kokkinakis, G.: An unconstrained handwritten recognition system. *International Journal on Document Analysis and Recognition*, IJDAR 4(4), 226–242 (2002)
14. Macé, S., Anquetil, E., Coüasnon, B.: A generic method to design pen-based systems for structured document composition: Development of a musical score editor. In: *Proceedings of the First Workshop on Improving and Assessing Pen-Based Input Techniques*, Edinburgh, pp. 15–22 (September 2005)
15. Mitobe, Y., Miyao, H., Maruyama, M.: A fast HMM algorithm based on stroke lengths for on-line recognition of handwritten music scores. In: *Proceedings of the ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)*. IEEE Computer Society, Los Alamitos (2004)
16. Miyao, H., Maruyama, M.: An online handwritten music symbol recognition system. *International Journal on Document Analysis and Recognition*, IJDAR 9(1), 49–58 (2006)
17. Ng, K.C., Boyle, R.D.: Recognition and reconstruction of primitives in music scores. *Image and Vision Computing* 14(1), 39–46 (1996)
18. Pérez, O., Piccardi, M., García, J.: Comparison between genetic algorithms and the baum-welch algorithm in learning hMMs for human activity classification. In: Giacobini, M. (ed.) *EvoWorkshops 2007. LNCS*, vol. 4448, pp. 399–406. Springer, Heidelberg (2007)
19. Sutherland, I.E.: Sketchpad: a man-machine graphical communication system. In: *Annual ACM IEEE Design Automation Conference*, pp. 507–524. ACM Press, New York (1988)
20. Xiaohui, H., Yuhui, S., Eberhart, R.: Recent advances in particle swarm. In: *Proceedings of the IEEE Congress on Evolutionary Computation*, Oregon, USA, pp. 90–96 (2004)
21. Xue, L., Yin, J., Ji, Z., Jiang, L.: A Particle Swarm Optimization for Hidden Markov Model Training. In: *Proceedings of the 8th International Conference on Signal Processing*, November 16–20 (2006)
22. Zhang, C., Shao, H., Li, Y.: Particle swarm optimisation for evolving artificial neural network. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 2487–2490 (2000)

Modeling Pheromone Dispensers Using Genetic Programming

Eva Alfaro-Cid¹, Anna I. Esparcia-Alcázar¹, Pilar Moya²,
Beatriu Femenia-Ferrer², Ken Sharman¹, and J.J. Merelo³

¹ Instituto Tecnológico de Informática, Universidad Politécnica de Valencia
{evalfaro,anna,ken}@iti.upv.es

² Instituto Agroforestal del Mediterráneo - Centro de Ecología Química Agrícola (IAM-CEQA), Universidad Politécnica de Valencia
{mmoyasa,beafefer}@ceqa.upv.es

³ Dept. de Arquitectura y Tecnología de Computadores, Universidad de Granada
jmerelo@geneura.ugr.es

Abstract. Mating disruption is an agricultural technique that intends to substitute the use of insecticides for pest control. This technique consists of the diffusion of large amounts of sexual pheromone, so that the males are confused and mating is disrupted. Pheromones are released using devices called dispensers. The speed of release is, generally, a function of time and atmospheric conditions such as temperature and humidity. One of the objectives in the design of the dispensers is to minimise the effect of atmospheric conditions in the performance of the dispenser. With this objective, the Centro de Ecología Química Agrícola (CEQA) has designed an experimental dispenser that aims to compete with the dispensers already in the market. The hypothesis we want to validate (and which is based on experimental results) is that the performance of the CEQA dispenser is independent of the atmospheric conditions, as opposed to the most widely used commercial dispenser, Isomate CPlus. This was done using a genetic programming (GP) algorithm. GP evolved functions able to describe the performance of both dispensers and that support the initial hypothesis.

1 Introduction

Codling moth, *Cydia pomonella* (L.) (Lepidoptera: Tortricidae), is the most important pest of pear and apple trees worldwide [1]. Chemical control programmes involve repeated applications of insecticides during the activity period of adults and larvae. This heavy pesticide programme is expensive and disruptive to beneficial orchard fauna, requiring, in addition, the supplemental application of spraying against secondary pests. Resistance to these insecticides, the possibility of cross-resistance to others and the limited number of products registered against codling moth, are motivating the implementation of pheromone-based Integrated Pest Management (IPM) systems. In these programmes, the mating disruption technique (MD) is the most promising tactic used to control this key pest in pome fruit production areas around the world [2,3,4].

One of the main factors contributing to the success of the MD technique is the right diffusion of the pheromone in the treated area; delivery of effective amounts of pheromone as well as a good distribution is essential [5]. Although several types of dispensers are commercially available, many of them suffer from a variety of problems. In some dispensers, pheromone load decreases very rapidly and, as a consequence, dispenser field-life is not long enough to cover the flight periods of the pest. Also, most devices are not biodegradable (plastic discs, polyethylene tubes and rubber septa, etc.), which causes an accumulation of polymeric materials in the treated environment. These limitations inspired the search for environmentally safe materials that could be used as controlled-release delivery systems for pheromones.

The Centro de Ecología Química Agrícola (CEQA) has developed a controlled-release pheromone dispenser based on its own technology which works effectively during the whole growing period. In addition, the dispenser is eco-friendly thanks to its biodegradable building materials. This characteristic makes it unique in the long-lasting pheromone dispenser world. Also, comparative studies of the release kinetics in the laboratory (under constant conditions of temperature and wind speed) and under field conditions seem to indicate that the experimental dispenser shows small sensitivity to climatic conditions. Thus, based on these preliminary data, the objective of this work was to demonstrate that the experimental dispenser sensitivity level to climatic conditions is lower than that shown by the most widely used commercial dispenser currently employed for controlling codling moth, the Isomate CPlus dispenser manufactured by Shin-Etsu.

To do so, we modeled the release kinetics of both dispensers through analytical functions obtained using genetic programming (GP)[6]. GP allows the evolution of functions, which makes it a very appropriate technique for modeling and regression problems. In the literature, numerous references can be found where GP is applied to these kind of problems [7,8]. Also, there are some references to the application of GP to agriculture [9], but they are scarce.

The choice of GP is attractive for a number of reasons including the associated ease of implementation, the lack of *a priori* model assumptions and its demonstrated ability for producing good results in real world problems [10]. In addition, GP offers an escape from the black box modeling techniques, providing an analytical function as a result. This is very relevant in problems where there is no knowledge of which are the most relevant variables for building the model, like the one we are dealing with in this paper. GP performs a natural selection of the most important inputs and also, the analysis of the final solution allows us to see which variables have been used and how they influence the final result.

In this work we contemplated three possibilities: time as the only variable, the inclusion of average daily values of temperature and humidity and the inclusion of maximum daily values of temperature and humidity. The results obtained in the three cases are compared using statistical analysis.

The rest of the paper is as follows. Section 2 describes the collection of data; Section 3 presents the problem to solve; and Section 4 shows the results obtained. Finally, in Section 5 some conclusions and future lines of work are discussed.

2 Data Collection

The dispensers (CEQA and Isomate CPlus) were placed at the start of the season (beginning of April) in a plot of 10 Hectares of surface of apple and pear trees in the region of Pla d'Urgell (Lleida, Spain). This plot was used to perform an efficiency test of the mating disruption technique using the experimental dispensers. Regularly, some dispensers (four dispensers of each type for each measurement of the residual) were retrieved and analysed for measuring the amount of pheromone remaining in the dispenser. It was quantified by gas-liquid chromatography with an internal standard.

We studied two series of data, one for the year 2005 and the other for the year 2006. Each of them covers a time frame of around 190 days, from the beginning of April to the end of September.

The available measurements of the residual are scarce: 15 values in 2005 and 7 in 2006 for the CEQA dispenser and 13 values in 2005 and 7 in 2006 for the CPlus dispenser. Each point represents the average pheromone residual of 4 dispensers.

The atmospheric data were obtained at the meteorologic station of El Poal (Lleida, Spain) through the Xarxa Agrometeorològica de Catalunya service. Our database includes average, minimum and maximum data of temperature and humidity for the season period of 2005 and 2006. We have worked exclusively with daily values of each variable, although it would be feasible to obtain these data every 30 minutes.

3 Problem Description

Given a dispenser that releases pheromone at a rate ν , find a function $\nu = \nu(t, T, H)$ where t represents the time, T the temperature and H the humidity. Alternatively, since it is nearly impossible from a practical point of view to measure the release rate ν , the residual r will be used instead. The residual is the percentage of product that has not been released. The release rate can be calculated as a function of the residual.

The available data are a sequence of points (r, t, T, H) obtained in field conditions. The objective is to find the relationship between r (and, consequently, ν) and the time, the temperature and the humidity. Our hypothesis is that the performance of the dispenser designed by the CEQA presents little or no dependency on atmospheric conditions and, therefore, $r \simeq r(t)$. However, that is not the case for commercial dispensers. Using GP we aim to obtain evidence that support this hypothesis.

Since measuring r is costly, there are very few measurements available and they are not equally spaced in time (there are more measurements initially, when the speed of emission is faster). But we cannot ignore the points where there is no residual measurement available, $(-, t, T, H)$, because the fluctuations of temperature and humidity could affect future values of r .

The problem to solve is then to find the function $r = r(t, T, H)$ given a set of experimental points (r, t, T, H) or $(-, t, T, H)$.

For solving the problem we used a GP algorithm implemented in ECJ [11]. It is a generational algorithm that uses elitism (0.1%). A population of 2000 individuals was evolved during 51 generations. The initialization method was ramped half and half. Selection was done with tournament of size 7. New populations were built using biased subtree crossover (80%), mutation (10%) and replacement (10%). Every experiment was run 10 times.

The choice of functions and terminals determines the search space of the algorithm. As terminals we included the time, the temperature, the humidity and an ephemeral random constant. The time variable is an integer in the interval $[0, t_{max}]$, where t_{max} is the number of days the experiment lasted (around 190). When a temperature or humidity terminal is included in a GP tree, a random integer n between 0 and 9 is generated. The value of n indicates if the temperature (or humidity) value is that of the current day (if $n = 0$) or of a previous day (if $n = 1$ it would be the previous day; if $n = 2$ it would be two days before, etc.). This way, the model can take into account the temperature and humidity conditions in days prior to the measurement of residual. The chosen function set includes the four arithmetic operators, the exponential and the logarithm.

In order to avoid that a resulting function performs operations that make no sense from a physical point of view, we implemented a strongly typed GP algorithm[12]. We defined four types: temperature, humidity, time and real, and every function was assigned the types it could accept as inputs and the type returned. For example, addition allows two inputs of the type temperature, but not an input of type temperature and a second input of type humidity. If the function adds up two values of type temperature then it returns a type temperature; if it adds up two values of type humidity then it returns a type humidity.

The cost function to minimise is the mean squared error (MSE): $MSE = \frac{1}{N} \sum_{k=1}^N (r_k^* - r_k)^2$, where N represents the number of available measurements, r_k^* is the value that function $r(\cdot)$ takes at point k and r_k is the experimentally measured residual at point k .

To asses that the resulting GP tree models accurately the dispenser performance, it needs to be validated with a set of values different from the one used for the construction of the model. In this case we used the data from year 2005 to build the model and those of year 2006 for its validation.

4 Experimental Results

In this section the results obtained with both dispensers are shown. For each dispenser three sets of 10 runs were executed. In the first set the terminal set included mean daily temperature, mean daily humidity and time as variables. In the second set, the terminal set included maximum daily temperature, maximum daily humidity and time as variables. Finally, in the last set of experiments time was the only variable considered. The best results obtained for each of these sets of 10 runs are shown together with a comparison study of the three approaches.

4.1 Dispenser CEQA

In a first approach we used mean values of temperature and humidity together with time to evolve a function that adjusted the performance of the CEQA dispenser. The optimised function that achieved the best result is shown below. If the tree is simplified, it can be represented as the product of two logarithms plus a constant (see Eq. 1). The function has 9 variables: time (t), 3 values of temperature, T_1 , T_2 y T_9 , and five values of humidity, H_1 , H_3 , H_5 , H_6 y H_7 , that is, $r = r(t, T_1, T_2, T_9, H_1, H_3, H_5, H_6, H_7)$.

$$r = 64.27 + \log \left(\frac{\frac{\exp(H_7)}{\exp(H_3)} - \frac{T_2^3 H_3^2 t^3}{H_1^3}}{H_5} + \frac{T_1 T_2 H_5 H_6 t^2}{H_1 H_3} \right) \log \left(\frac{80.97 A}{t^2} \right) \quad (1)$$

where $A = \log \left(T_1 t \left(T_2 + \frac{t^3}{\log(T_2)} \right) \right) + T_9$

Since it is not known whether the most relevant atmospheric factors are the average temperature and humidity or the maximum values of temperature and humidity, the same experiment was run using maximum (instead of mean) daily values of temperature and humidity as terminals. In this case, the cost assigned to the best result is slightly better than that obtained by the best result using mean values of temperature and humidity.

The resulting function consist of the addition of three terms: a constant plus other two terms that nest several exponential and logarithmic operations. In this case, the residual, $r = r(t, T_0, T_2, T_3, T_4, T_7, T_8)$, is independent of humidity.

$$r = 65.01 - \frac{t \log T_0}{T_4} \log \left(2.7 + \frac{9.12}{t} \right) - \frac{T_4 \log(0.01t)}{\log \left(\log \left(\frac{-0.01 T_2 T_4^2 \exp(\frac{9.12}{t})}{A} \right) \right)} \quad (2)$$

where $A = \log \left(\frac{T_3^2 T_8}{64.05(T_2 + T_7)t} \right) \log \left(\frac{3.58(65.01 - t \log(T_0))}{T_2} \right)$

Ideally, the performance of the dispenser should not depend on the atmospheric conditions. So we are interested in studying what happens if we remove temperature and humidity from the terminal set, and time is the only variable considered.

In this case the best result is obtained by a very simple function. It consist of the addition of 5 terms: a constant, a logarithm and other three terms that are functions of time and the logarithm of time.

$$r = 96.03 - \log \left(\frac{t^2}{(79.47 - t)(70.77 - t)} \right) - \frac{t}{4.41} - \frac{t}{\log(t)} + \frac{t}{679.29 - 4t} \quad (3)$$

The function presents four discontinuities: $t = 79.47$, $t = 70.77$, $t = 679.29/4 = 169.82$ y $t = 1$ ($\log(1) = 0$). To avoid the generation of solutions that are not

computable the division operator has a protection mechanism that returns 1 in case of a division by 0. Anyway, in this case the mechanism is only activated in $t = 1$ because the time is measured in days and, therefore, it is always an integer.

Next we present the statistical analysis of the results obtained for the CEQA dispenser in the validation process. The models obtained using data of the year 2005 were validated using data of year 2006. We performed a non-parametrical statistical test (Kruskal-Wallis) on the validation results in order to determine if the differences observed are significant. The choice of a non-parametrical test was motivated by the non-normality and non-homoskedasticity of the samples.

Figure 1 presents the boxplots for the validation results. The figure shows three boxes. The left-hand box presents the validation results of the models built using mean daily temperature, mean daily humidity and time variables; the box in the centre presents the validation results of the models built using maximum daily temperature, maximum daily humidity and time variables; and, finally, the right-hand box presents the validation results of the models built using only time as a variable. In the figures we can see a box and whisker plot for each column of data. The boxes have lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the boxes to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers. They are represented with a plus symbol. The notches in a box plot represent a robust estimate of the uncertainty about the medians for box-to-box comparison. Boxes whose notches do not overlap indicate that the medians of the two groups differ at the 5% significance level. To make easier the interpretation of the figure we did not plot one of the outliers of the left box. It was more than eight times bigger than the rest of the values and it distorted the figure.

The statistical test concludes that the results obtained using average values of temperature and humidity and those obtained using maximum values instead are not significantly different. On the other hand, the numerical costs obtained using only time as a terminal are significatively better than the other two. The confidence level is higher than 99.9% when they are compared against the results obtained using the maximum values of temperature and humidity; and it is higher than 99.95% when compared against the result obtained using average values of temperature and humidity. The inclusion of atmospheric data does not contribute to improve the fitting of the function; on the contrary, the opposite seems to be true: it only complicates the search space preventing the GP algorithm from finding better solutions. Thus, although this analysis does not allow us to conclude that the performance of the CEQA dispenser is not influenced by the temperature and humidity conditions, the statistical tests confirms the superiority of the results obtained using only time as a variable. This evidence validates our hypothesis of the independence of the release kinetics of the pheromone dispenser designed by the CEQA with respect to the atmospheric conditions.

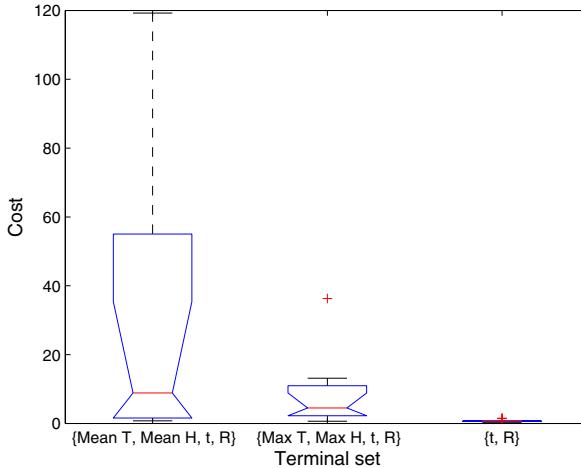


Fig. 1. Validation results obtained for each set of experiments - dispenser CEQA

4.2 Dispenser CPlus

In this section we present the results that we obtained by performing a parallel analysis to the commercial dispenser CPlus. When using the mean daily values of temperature and humidity the function that provides the best results is as follows:

$$r = 64.5 + \frac{55.08}{\log(\log(H_7))} - \frac{0.11H_9 t}{\log(T_7) \log(H_9)} + \frac{36.31}{t \exp\left(\frac{36.31}{2tH_9 \exp\left(\frac{-T_7 t}{18.49 \log(H_7)}\right)}\right)} \quad (4)$$

As it can be seen, it consists of 4 terms: a constant value, a second term that is a function of the logarithm of H_7 , a third term that is a function of time, H_9 and T_7 and, a final term where two exponential functions are nested. In this case, $r = r(t, T_7, H_7, H_9)$.

As with the CEQA dispenser, the experiments were repeated using maximum daily values of temperature and humidity. The numerical cost values obtained were slightly better than when using average values. The best resulting function is more complex than in previous cases. Basically it consists of a constant value plus a very complex logarithmical term in which products, divisions, logarithms and exponentials are nested. According to this function the residual depends fundamentally on time and temperature values. There is only one humidity value in the expression and it is placed in a position where it does not influence much the final result, therefore we could consider that the function is independent of humidity. Thus, $r = r(t, T_0, T_1, T_2, T_7, T_9, H_9)$.

$$r = 92.29 + \frac{t}{1.42T_9} \log \left(\frac{A}{-1506.74t \log \left(\frac{A}{t^2} \right) \log \left(\frac{\exp \left(\frac{179.87}{t} \right)}{-75.79T_2 \exp \left(\frac{-t^2}{43.93T_7} \right) B} \right)} \right) \quad (5)$$

where $A = \frac{\exp \left(\frac{271.53}{t} \right)}{T_7 \log \left(\frac{(T_0+T_1)t}{T_1} \right)}$

$$B = \log(83.3t) \log \left(\frac{271.53}{t} \right) \log \left(\frac{T_1}{H_9 \exp \left(\frac{271.53}{t} \right)} \right) t$$

Finally, we considered the case where time is the only variable available for the GP algorithm in order to see if atmospheric conditions are relevant when modeling the performance of the CPlus dispenser. The expression yielding the best fitting is quite complex (see Eq. 6). It consists of 4 terms: the first term is a constant value, the second term is the product of time and a constant value, the third term is a logarithmic term that depends on time, the inverse of time and the logarithm of time and, the fourth term is a fraction where additions, products and logarithms are combined.

$$r = \frac{329.56 - t}{4.31} + \log \left(\frac{74.32}{t} + t - 93.95 + \log(t) \right) + \frac{N}{\log(81.46 - 1.29t - D)} \quad (6)$$

where $N = 81.36 - 1.29t - \frac{5642.08}{M(\log(t) - 71.6(74.93 - \log(\log(t))(t - 83.38)))}$

$$D = \frac{71.6(79.8 - \log(2 \log(t)))}{\log \left(\frac{56.67}{t} (\log(t) - 74.93(74.93 - \log(t)(t - 83.38))) \right)}$$

$$M = \log(\log(t)(0.77t - 15.67)(0.02t - 1.47))$$

We performed a non-parametrical statistical test (Kruskal-Wallis) for the validation results obtained for CPlus; the resulting boxplot is shown in Fig. 2. As in Fig. 1, the left-hand box presents the validation results of the models built using mean daily temperature, mean daily humidity and time variables; the box in the centre presents the validation results of the models built using maximum daily temperature, maximum daily humidity and time variables; and, finally, the right-hand box presents the validation results of the models built using only time as a variable.

The statistical test concludes that the results obtained using the maximum values of temperature and humidity are significantly better than any of the other two with a confidence level of 99.95%. The results obtained using average values of temperature and humidity and those obtained using only a time variable are equivalent from a statistical point of view. Therefore, in the case of the commercial dispenser CPlus, the results indicate that there is a clear influence of the atmospheric conditions in the performance of the dispenser.

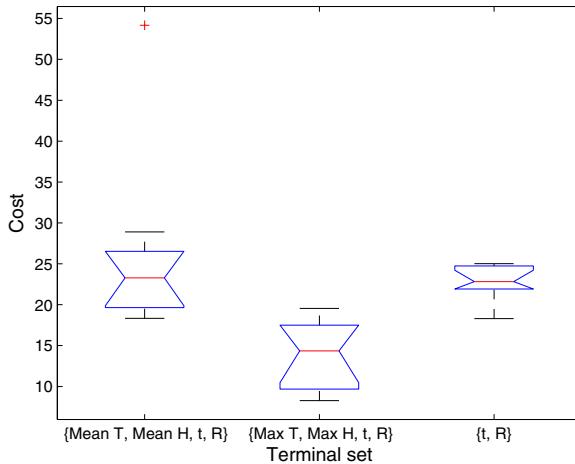


Fig. 2. Validation results obtained for each set of experiments - dispenser CPlus

5 Conclusions and Further Work

Genetic programming has proven to be capable of finding functions that fit well the performance of both dispensers: the experimental dispenser CEQA and the commercial dispenser CPlus.

Regarding the influence of the atmospheric conditions in the performance of both dispensers, the results show that for the CEQA dispenser the fitting of the functions is better when the only variable under consideration is time. This seems to indicate that the inclusion of temperature and humidity data only adds noise and complicates the search space. Although this is not a conclusive proof of the independence of the pheromone residual from the atmospheric conditions, it can be considered as an evidence in that sense.

On the other hand, the conclusions of the statistical test performed to the results obtained with the data of CPlus dispenser are very different. In this case the test reveals that there is a significant difference between the results obtained using maximum values of temperature and humidity and the rest. The functions evolved using maximum values of temperature and humidity model the performance of the dispenser better than any other. This confirms prior experimental evidence that the atmospheric conditions have a big influence in the performance of these dispensers.

Another observation that can be deducted from the analysis of the resulting functions is that the best functions obtained with maximum values of temperature and humidity for both dispensers are independent (or nearly independent) of the humidity values. It seems that the daily maximum temperature is a more relevant factor than the daily maximum humidity.

As future work we plan to model the release of pheromone in the environment. This is a question of great interest, especially from an economic point of view,

since it would allow the optimisation of the placement of dispensers in the plot, hence minimising the number of dispensers needed to guarantee an efficient pest control.

Another aspect we want to consider is the inclusion of the gradient of temperature as a terminal for the GP algorithm, as it may be the case that the dispensers are more sensitive to sharp changes of temperature than to the temperature itself.

Acknowledgements

This work was partially funded by project NoHNES (Spanish Ministerio de Educación y Ciencia - TIN2007-68083-C02).

References

- Witzgall, P., Bengtsson, M., Rauscher, S., Liblikas, I., Bäckman, A.C., Coracini, M., Anderson, P., Löfqvist, J.: Identification of further sex pheromones in the codling moth, *cydia pomonella*. *Entomologia Experimentalis et Applicata* 101, 131–141 (2001)
- Barnes, M., Millar, J., Kirsch, P., Hawks, D.: Codling moth (lepidoptera: Tortricidae) control by dissemination of synthetic female sex pheromone. *Journal of Economic Entomology* 85, 1274–1277 (1992)
- Cardé, R., Minks, A.: Control of moth pest by mating disruption: successes and constraints. *Annual Review of Entomology* 40, 559–585 (1995)
- Witzgall, P., Stelinski, L., Gut, L., Thomson, D.: Codling moth management and chemical ecology. *Annual Review of Entomology* 53, 503–522 (2008)
- Welter, S., Pickel, C., Millar, J., Cave, F., Van Steenwyk, R., Dunley, J.: Pheromone mating disruption offer selective management options for key pest. *California Agriculture* 59, 16–22 (2005)
- Koza, J.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
- Keijzer, M.: Scaled symbolic regression. *Genetic Programming and Evolvable Machines* 5, 259–269 (2004)
- Gustafson, S., Burke, E., Krasnogor, N.: On improving genetic programming for symbolic regression. In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 912–919. IEEE Press, Los Alamitos (2005)
- Gutiérrez, P., López-Granados, F., Peña Barragán, J.M., Jurado-Expósito, M., Hervás-Martínez, C.: Logistic regression product-unit neural networks for mapping *ridolfia segetum* infestations in sunflower crop using multitemporal remote sensed data. *Computers and Electronics in Agriculture* 64, 293–306 (2008)
- Kotanchek, M., Smits, G., Kordon, A.: Industrial strength genetic programming. In: *Genetic Programming Theory and Practice*, pp. 239–255. Kluwer Academic, Dordrecht (2003)
- EClab - George Mason University (ECJ),
<http://cs.gmu.edu/~eclab/projects/ecj>
- Montana, D.J.: Strongly typed genetic programming. *Evolutionary Computation* 3, 199–230 (1995)
- Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press (1975)

NK Landscapes Difficulty and Negative Slope Coefficient: How Sampling Influences the Results

Leonardo Vanneschi¹, Sébastien Verel², Marco Tomassini³, and Philippe Collard²

¹ Dipartimento di Informatica, Sistemistica e Comunicazione (D.I.S.Co.),
University of Milano-Bicocca, Milan, Italy

² I3S Laboratory, University of Nice–Sophia Antipolis, France

³ Information Systems Department, HEC, University of Lausanne, Lausanne, Switzerland

Abstract. Negative Slope Coefficient is an indicator of problem hardness that has been introduced in 2004 and that has returned promising results on a large set of problems. It is based on the concept of fitness cloud and works by partitioning the cloud into a number of bins representing as many different regions of the fitness landscape. The measure is calculated by joining the bins centroids by segments and summing all their negative slopes. In this paper, for the first time, we point out a potential problem of the Negative Slope Coefficient: we study its value for different instances of the well known NK-landscapes and we show how this indicator is dramatically influenced by the minimum number of points contained in a bin. Successively, we formally justify this behavior of the Negative Slope Coefficient and we discuss pros and cons of this measure.

1 Introduction

The intrinsic difficulty of searching a given problem space can be characterized in a statistical way and there have been several attempts in the last fifteen years of using algorithm-independent statistical measures of fitness landscapes with this aim. Among these, we may mention Fitness-Distance Correlation (FDC) analysis [2], Landscape Correlation Functions [13], Density of States [7], Negative Slope Coefficient (NSC) [9,10], and several others (see [3] for general discussions). The goal of all these methods is to characterize the hardness of a given problem space with a single number in such a way that landscapes that are classified as “difficult” will resist solutions or request a large amount of computational resources to any algorithm or heuristic, and those that are “easy” will be solvable with limited computational resources. Clearly, if one such method would give reliable results, it would be very useful in order to compare problems and problem classes among themselves. If possible, the method should be “predictive” in the sense that it should be possible to apply it to an unknown fitness landscape without the need to know any particular feature of the landscape, such as number and position of optima, in advance. For example, FDC is not such a predictive measure, as the position of the global optimum (or at least of a good sub-optimum if the latter is unknown) is required to perform the analysis. Correlation Analysis, Density of States, and NSC calculations do not need any previous knowledge instead. Some of these measures have proved useful as they give a global view of the hardness of a given

problem or problem class, and they can help suggest the use of particular neighborhoods and search techniques as being more appropriate than others to search a given landscape. However, by their very nature, many of the above measures are based on statistics, i.e. except for the simplest and smallest landscapes, one needs to sample the landscape in some way and thus the results are always an approximation of the true situation. Because of this aspect, these approaches can all be deceived by particular features of the landscape, natural or artificially introduced [3].

In this paper we study the NSC difficulty measure and we show some of its advantages and weaknesses on a particular class of fitness landscapes: the *NK* family.

In the next section we briefly define NSC and how it is computed. In Section 3 we recall the definition of the *NK* landscapes. In Section 4 we numerically examine the behavior of NSC on these landscapes and discuss the results in Section 5, offering a theoretical interpretation of them. Finally, Section 6 concludes this work.

2 Negative Slope Coefficient

Evolvability quantifies the ability of genetic operators to improve fitness quality. One possible way to study it is the *fitness cloud* introduced in [12]. The fitness cloud relative to the local search operator *op* is the conditional bivariate probability density $P_{op}(Y = \tilde{\phi} | X = \phi)$ of reaching a solution of fitness value $\tilde{\phi}$ from a solution of fitness value ϕ applying the operator *op*. To visualize the fitness cloud in two dimensions, we plot the scatterplot $FC = \{(\phi, \tilde{\phi}) | P_{op}(\phi, \tilde{\phi}) \neq 0\}$.

Since high-fitness points tend to be much more important than low-fitness ones in determining the behaviour of evolutionary algorithms, an alternative algorithm to generate fitness clouds was proposed in [10]. The main steps of this algorithm can be informally summarised as follows: (1) Generate a set of individuals $\Gamma = \{\gamma_1, \dots, \gamma_n\}$ by sampling the search space and let $f_i = f(\gamma_i)$, where $f(\cdot)$ is the fitness function. (2) For each $\gamma_j \in \Gamma$ generate k neighbours, v_1^j, \dots, v_k^j , by applying a genetic operator to γ_j and let $f'_j = \max_j f(v_j)$. (3) Finally, take $C = \{(f_1, f'_1), \dots, (f_n, f'_n)\}$ as the fitness cloud.

This definition clearly leaves at least three unanswered questions: (i) how do we sample the search space to generate abscissas? (ii) what variation operator do we use to generate neighbors? (iii) how many neighbors do we generate for each individual? The answer to question (i) has been discussed in [11], where the Metropolis algorithm [5] has been proposed to sample the search space. To answer questions (ii) and (iii) we have followed [8]: we have used a single bit flip mutation to generate neighbors and tournament selection with tournament size equal to 2 has been used to sample neighborhoods.

The fitness cloud can be of help in determining some characteristics of the fitness landscape related to evolvability and problem difficulty. But the mere observation of the scatterplot is not sufficient to quantify these features. The Negative Slope Coefficient (NSC) has been defined to capture with a single number some interesting characteristics of fitness clouds. It can be calculated as follows: let us partition C into a certain number of separate ordered “bins” C_1, \dots, C_m such that $(f_a, f'_a) \in C_j$ and $(f_b, f'_b) \in C_k$ with $j < k$ implies $f_a < f_b$ [11]. Next we consider the average fitnesses $\bar{f}_i = \frac{1}{|C_i|} \sum_{(f, f') \in C_i} f$ and $\bar{f}'_i = \frac{1}{|C_i|} \sum_{(f, f') \in C_i} f'$. The points (\bar{f}_i, \bar{f}'_i) can be seen as the vertices of a polyline, which effectively represents the “skeleton” of the fitness cloud. For

each of the segments of this we can define a *slope*, $S_i = (f'_{i+1} - f'_i)/(f_{i+1} - f_i)$. Finally, the negative slope coefficient is defined as $NSC = \sum_{i=1}^{m-1} \min(0, S_i)$.

The hypothesis proposed in [9,10,11] is that NSC should classify problems in the following way: if $NSC = 0$, the problem is easy; if $NSC < 0$ the problem is difficult and the value of NSC quantifies this difficulty: the smaller its value, the more difficult the problem. The justification put forward for this hypothesis was that the presence of a segment with negative slope would indicate a bad evolvability for individuals having fitness values contained in that segment as neighbours would be, on average, worse than their parents in that segment [9]. A more formal justification for NSC has been given in [6].

The results reported in [10,11,6,8] confirmed that the NSC is a suitable hardness indicator for many well known GP and GA benchmarks and synthetic problems, including various versions of the symbolic regression problem, the even parity problem of many different orders, the artificial ant problem on the Santa Fe trail, the multiplexer problem, the intertwined spirals, the GP Trap Functions, Royal Trees, the Max problem, Onemax, Trap, and Onemix, which naturally capture some typical features of easy and difficult fitness landscapes, and the SAT problem.

In [11] a method called size-driven bisection has been proposed to partition the cloud into bins. This method (used also in this paper) has some degrees of freedom, like for instance the minimum number of points that can be contained in a bin. In this paper we show how the NSC value is dramatically influenced by this parameter.

3 The NK-Landscapes

The *NK* family of landscapes [4] is a problem-independent model for constructing multimodal landscapes that can gradually be tuned from smooth to rugged. In the model, N refers to the number of (binary) genes in the genotype (i.e. the string length) and K to the number of genes that influence a particular gene (the epistatic interactions). By increasing the value of K from 0 to $N - 1$, *NK* landscapes can be tuned from smooth to rugged. The fitness function of a *NK* landscape is a function $f_{NK} : \{0, 1\}^N \rightarrow [0, 1]$ defined on binary strings with N bits. An 'atom' with fixed epistasis level is represented by a fitness component $f_i : \{0, 1\}^{K+1} \rightarrow [0, 1]$ associated to each bit i . Its value depends on the allele at bit i and also on the alleles at the K other epistatic positions. (K must fall between 0 and $N - 1$). The fitness $f_{NK}(s)$ of $s \in \{0, 1\}^N$ is the average of the values of the N fitness components f_i : $f_{NK}(x) = \frac{1}{N} \sum_{i=1}^N f_i(s_i, s_{i_1}, \dots, s_{i_K})$, where $\{i_1, \dots, i_K\} \subset \{1, \dots, i-1, i+1, \dots, N\}$. Many ways have been proposed to choose the K other bits from N bits in the bit string. Two possibilities are mainly used: adjacent and random neighborhoods. With an adjacent neighborhood, the K bits nearest to the bit i are chosen (the genotype is taken to have periodic boundaries). With a random neighborhood, the K bits are chosen randomly on the bit string. Adjacent neighborhood is used in this paper. Each fitness component f_i is specified by extension, *i.e.* a number $y_{s_i, s_{i_1}, \dots, s_{i_K}}^i$ from $[0, 1]$ is associated with each element $(s_i, s_{i_1}, \dots, s_{i_K})$ from $\{0, 1\}^{K+1}$. Those numbers are uniformly distributed in the range $[0, 1)$.

4 Experimental Results

Table 1 presents the results that we have obtained studying in an exhaustive way the search space with $N = 20$, nine different values of K ($K = 2, 4, 6, 8, 10, 12, 14, 16, 18$) and three different values for the minimum number of points that a bin can contain (this value has been indicated with y in the table and the values used are 50, 500 and 5000). Results have been obtained by running 50 independent executions of the algorithm that calculates the NSC. We point out that, even though we have exhaustively studied the whole search space, the neighbors of each point (ordinates of the scatterplot) are obtained by tournament selection and thus we do not have the same fitness cloud for each different execution. Results reported in the table are the average values of the NSC with their standard deviations over these 50 independent executions. Even though for each value of y the NSC becomes more negative as K grows (which is reasonable and seems to indicate that the NSC correctly predicts the difficulty of the different problem instances for that value of y), we also remark that NSC gets closer to zero as y increases. Even more importantly for $y = 5000$ the NSC is equal to zero for $K = 2, 4, 6, 8, 10, 12$, but these problem instances clearly have different difficulties [4,1,12]¹.

Table 1. Results obtained studying in an exhaustive way the search space for $N = 20$, 9 different values of K and 3 different values for the minimum number of points per bin (y in table). Values in the table indicate the average NSC over 50 independent executions, with their respective standard deviations (indicated between parenthesis under the value).

NSC values for $N = 20$, Exhaustive Study									
y	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 10$	$K = 12$	$K = 14$	$K = 16$	$K = 18$
50	0.0 (0.0)	-28.4826 (7.7675)	-32.7045 (4.2704)	-34.1192 (3.5119)	-37.9779 (5.1591)	-42.5298 (4.5709)	-46.8483 (3.1358)	-50.1184 (7.5342)	-54.3703 (6.6453)
500	0.0 (0.0)	-12.7465 (5.7453)	-19.7342 (3.8563)	-21.8463 (2.8956)	-28.0463 (4.8764)	-33.9463 (3.0265)	-37.8465 (3.8654)	-41.7453 (5.9345)	-45.8465 (5.2367)
5000	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	-0.0139 (0.0286)	-0.06223 (0.0345)	-0.2458 (0.0745)

Table 2 presents the results obtained by sampling the search space with the Metropolis algorithm for $N = 20$ and the same values of K as in Table 1. This time, we have considered different values of the sample size (x in the table) and of the minimum number of points per bin (y in the table). In particular, we have used the following values: $x = 4,000$ and $y = 50$, $x = 40,000$ and $y = 500$ and $x = 400,000$ and $y = 5,000$. As in the case of Table 1, we have considered 50 independent executions of the algorithm that calculates the NSC and we have reported in table the average values of the NSC with their respective standard deviations. Even though quantitatively different, results

¹ The performances of an optimization heuristic (for instance Genetic Algorithms) on these problem instances could be reported here. Anyway, given that these performances have already been studied in literature (see for instance [4,1,12]), we do not report these results here to save space. However, to understand the concepts of this paper, it is only important to know that, by construction, the difficulty of the NK landscapes increases as K increases (given that increasing K we augment the ruggedness of the landscape).

Table 2. Results obtained sampling the search space with the Metropolis algorithm for $N = 20$, 9 different values of K and 3 different values for the sample size (x in table) and the minimum number of points per bin (y in table). Values in the table indicate the average NSC over 50 independent executions, with their respective standard deviations (indicated between parenthesis under the value).

NSC values for $N = 20$, Sampling									
(x, y)	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 10$	$K = 12$	$K = 14$	$K = 16$	$K = 18$
(4000, 50)	0.0 (0.0)	-18.6354 (4.8645)	-30.7453 (6.7856)	-38.8465 (7.6594)	-42.7594 (6.6403)	-47.9058 (5.9465)	-51.9556 (6.9453)	-65.8452 (7.9455)	-71.8562 (6.9786)
(40000, 500)	0.0 (0.0)	-1.075 (0.8645)	-5.4964 (2.4229)	-11.3245 (2.3212)	-13.6118 (1.1786)	-17.3116 (2.0642)	-18.9131 (4.2884)	-20.7859 (1.8944)	-23.3943 (3.7996)
(40000, 5000)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	-0.01416 (0.0332)	-0.0598 (0.0482)	-0.2206 (0.0822)

Table 3. Results obtained sampling the search space with the Metropolis algorithm for $N = 30$, 9 different values of K and 3 different values for the sample size (x in table) and the minimum number of points per bin (y in table). Values in the table indicate the average NSC over 50 independent executions, with their respective standard deviations (indicated between parenthesis under the value).

NSC values for $N = 30$, Sampling						
(x, y)	$K = 2$	$K = 6$	$K = 10$	$K = 14$	$K = 18$	$K = 22$
(4000, 50)	0.0 (0.0)	-62.3787 (6.9563)	-77.3495 (8.9543)	-88.3785 (8.9954)	-96.78344 (7.2855)	-102.4533 (8.3352)
(40000, 500)	0.0 (0.0)	-12.3129 (2.5587)	-15.1808 (4.5633)	-17.4112 (5.3856)	-23.2661 (4.5763)	-35.3933 (6.8544)
(40000, 5000)	0.0 (0.0)	0.0 (0.0)	-0.9191 (0.0547)	-1.4281 (0.0785)	-3.8037 (0.0649)	-5.9854 (0.0543)

in Table 2 are qualitatively similar to the ones in Table 1 in the sense that they lead us to similar conclusions: independently from the value of y , the NSC gets more negative as K grows, but the NSC is also closer to zero as y grows. As in Table 1, also in Table 2 we can see that the NSC is always equal to zero for $K = 2$ when $y = 50$ and $y = 500$, while NSC is equal to zero for $K = 2, 4, 6, 8, 10, 12$ when $y = 5000$. Given that what broadly makes the difference between an easy problem and a hard one for the NSC is the absence or presence of negative slopes respectively, we can say that the Metropolis sampling algorithm allows us to broadly classify problems as if the search space was studied exhaustively. This confirms the suitability of the used sampling method, in the sense that our interpretation of the results obtained using it is the same as the one for the exhaustive study.

We have also obtained analogous results with $N = 15$, but we do not report them here for lack of space. However, also in that case the NSC gets more negative as K increases and gets nearer to zero as y increases.

Table 3 presents the results that we have obtained on a larger search space, i.e. for $N = 30$. In this case, we have not been able to study the search space exhaustively, and

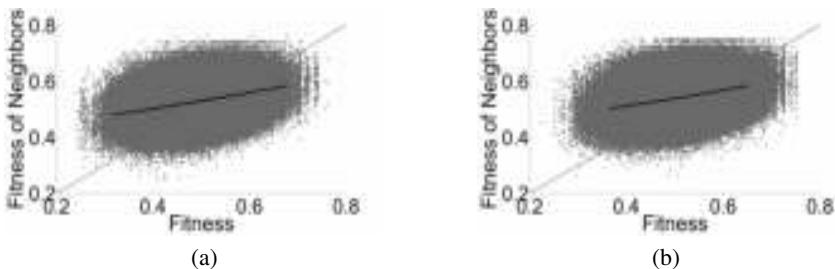


Fig. 1. Scatterplots and segments for the NSC calculation in two particular runs among the ones used for generating results in Table 1. In these experiments: $N = 20$, $K = 12$ and: (a): minimum number of points per bin = 50; (b): minimum number of points per bin = 5,000.

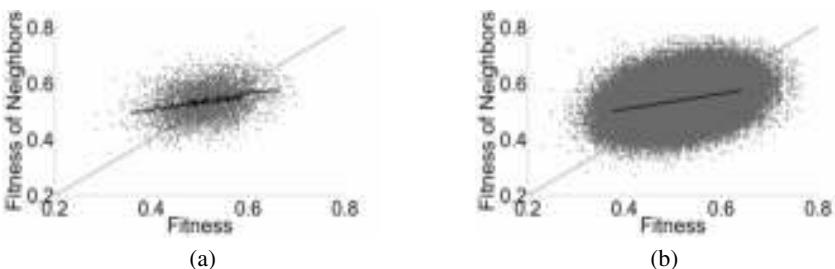


Fig. 2. Scatterplots and segments for the NSC calculation in two particular runs among the ones used for generating results in Table 2. In these experiments: $N = 20$, $K = 12$ and: (a): sample size = 4,000 and minimum number of points per bin = 50; (b): sample size = 400,000 and minimum number of points per bin = 5,000.

thus we only present the results that we have obtained using the Metropolis sampling. As for the other cases studied, the results that we present are the average values of the NSC with their standard deviations. We have used the same values of the sample size (x in the table) and of the minimum number of points per bin (y in the table) as in Table 2, i.e.: $x = 4,000$ and $y = 50$, $x = 40,000$ and $y = 500$ and $x = 400,000$ and $y = 5,000$. Also in this case, the NSC gets more negative as K increases and gets nearer to zero as y increases.

In Figure 1, we report the scatterplots and segments that have allowed us to calculate the NSC in two different executions for $N = 20$ and $K = 12$. The difference between these two scatterplots is in the value of the minimum number of points per bin y : $y = 50$ in Figure 1(a) and $y = 5,000$ in Figure 1(b). We have chosen to report the scatterplots of the executions that have returned the more similar NSC values to the average over the 50 independent runs that we have used in Table 1. Figure 1(a) shows some small (in abscissa) segments with strong negative slope. Figure 1(b) contains no segment with negative slope (in fact the NSC is zero for $N = 20$, $K = 12$ and $y = 5,000$).

Figure 2 reports the analogous results that we have obtained using the Metropolis sampling algorithm. We report the clouds of the run that has returned the NSC value closer to the average for the following cases: $N = 20$, $K = 12$ and the following values of the sample size (x) and the minimum number of points per bin (y): $x = 4,000$ and

$y = 50$ in Figure 2(a); and $x = 400,000$ and $y = 5,000$ in Figure 2(b). In this case the differences between the polylines of Figures 2(a) and 2(b) are more visible, but the results are qualitatively similar: many segments with negative slopes are present when $y = 50$ and none of them when $y = 5,000$.

5 Discussion

The previous section has pointed out pros and cons of the NSC. Synthetically the pros are: (i) The NSC gets more negative as K increases (and it is well known that NK landscapes get more rugged, and thus more difficult, as K increases). (ii) Even though quantitatively different, results obtained by sampling the search space with the Metropolis algorithm are qualitatively similar to the ones that we have obtained studying the search space in an exhaustive way, in the sense that they broadly classify instances into easy and hard in the same way. Drawbacks of the NSC pointed out by our experiments are: (a) Results dramatically change as the value of the minimum number of points (y) per bin changes. (b) For a high value of y (i.e. $y = 5000$), the NSC fails to correctly classify the hardness of some problem instances: for many different values of K the NSC is equal to zero, while the respective problem instances have different difficulties.

However, these NSC “drawbacks” were completely expected and we can exhibit a formal justification for them.

To explain them, we compute the distribution D (under some hypothesis) of the difference $f'_{i+1} - f'_i$ where for each i , f'_i is the ordinate of the centroid of bin i . Let B and B_δ be two bins where the centroids have respectively abscissa f and $f + \delta$ with $\delta > 0$. Let denote respectively F and F_δ the random variables which give the distribution of the maximum fitness from k random neighbor solutions. We suppose that F and F_δ are independent and follow respectively the normal law N of parameters m and σ , and the normal law N_δ of parameters $m + \delta'$ and $\alpha \cdot \sigma$. The averages of neighbor fitness m and $m + \delta'$ are estimated by a sample of n points. So, the estimators follow the laws $f' = \frac{1}{n} \sum_{i=1}^n N_i$ and $f'_\delta = \frac{1}{n} \sum_{i=1}^n N_{\delta,i}$ where each N_i follows N and each $N_{\delta,i}$ follows N_δ . If we make the approximation that the random variables from each sum are independent, then f' follows a normal law of parameters m and $\frac{\sigma}{\sqrt{n}}$ and f'_δ follows a normal law of parameters $m + \delta'$ and $\alpha \cdot \frac{\sigma}{\sqrt{n}}$. Furthermore, f' and f'_δ are supposed to be independent so the distribution D follows a normal law of parameters δ' and $\sigma_n = \sigma \sqrt{\frac{1+\alpha^2}{n}}$.

Then, the probability that the estimated slope has a negative value is the value of of repartition function of normal law D in 0, i.e.:

$$P(D < 0) = \frac{1}{2}(1 - \text{erf}(\frac{\delta'}{\sqrt{2}\sigma_n}))$$

where erf is the “error function” encountered in integrating the normal distribution. Furthermore, the average of the “negative slope” \bar{S} between B and B_δ is given by the average value of D/δ when D is negative, i.e.:

$$\bar{S} = \int_{-\infty}^0 z P(D = z) dz = \frac{\delta'}{\delta} \frac{1}{2} (1 + \text{erf}(\frac{-\delta'}{\sqrt{2}\sigma_n})) - \frac{\sigma_n}{\delta \sqrt{2\pi}} e^{\frac{-\delta'^2}{2\sigma_n^2}} \quad (1)$$

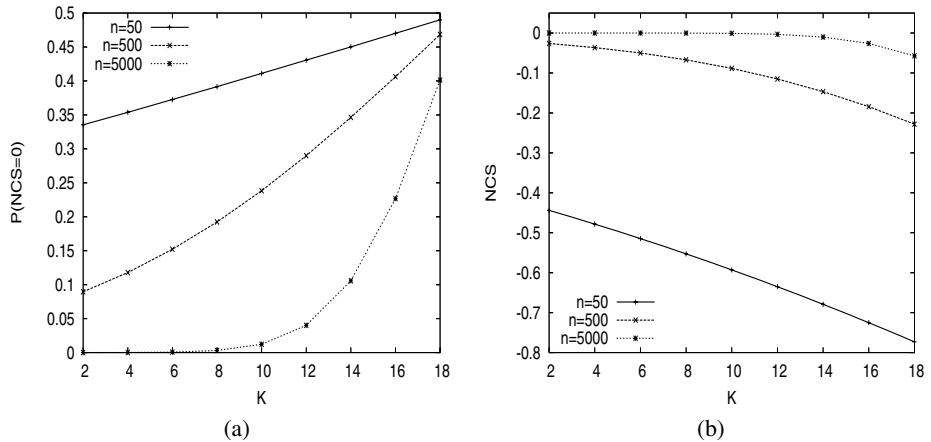


Fig. 3. (a): Theoretical probability of having a non-zero NSC value for various values of K . (b): Theoretical value of the NSC for one generic bin for various values of K . In these experiments: $N = 20$ and minimum number of points per bin 50, 500, 5,000.

Those equations show that if the number of points n increases, the value of σ_n decreases and then the probability of having a negative value of D tends to 0. Furthermore the value of the negative slope \bar{S} decreases and becomes 0 when the number of points tends to infinity. When the value of δ' is high, it is easier to increase the fitness in the neighborhood of solutions which have the fitness $f + \delta$ than when δ' is low and the higher is the value of δ' , the less difficult the problem is (as already pointed out in [12]). Equation (1) shows that the negative slope increases as δ' decreases. Thus, we can say that, as for the experimental results, also these theoretical results show that the negative slope decreases when the number of points per bin increases and that the average of the negative slope increases when the difficulty increases.

For NK landscapes, the average fitness in the neighborhood for a solution of fitness f is given by $(1 - \frac{K+1}{N})(f - 0.5) + 0.5$. Even if the fitness is obtained using tournament selection, the fraction $\frac{\delta'}{\delta}$ is equal to $1 - \frac{K+1}{N}$ (as shown in [12]). As pointed out before, for a δ given, δ' decreases as the difficulty increases. In the experiments of Section 4, we have that δ is approximately equal to 0.04 and σ is approximately equal to 0.05. Figure 3(a) shows the probability of having a negative value for the NSC increases. For a given K , the probability decreases when the number of points per bins increases. The theoretical values seem to be consistent with the experimental data, even though we have made the hypothesis that the fitness of neighbors is normally distributed and that the random variables are independent.

As a direct consequence of this theoretical study, we deduce that when it is possible to calculate the slopes with a linear regression (as explained in [12]), of the fitness cloud, it is not necessary to compute the NSC. In fact the slopes in the linear regression

have the same properties as the value of the NSC, but the errors of the estimation are clearly smaller. In some cases (for instance when Genetic Programming problems are considered), the fitness cloud cannot be calculated by a linear regression, and thus it makes sense to calculate the NSC to estimate it.

6 Conclusions and Future Work

NK landscapes are a very interesting test case for Negative Slope Coefficient (NSC). In fact, this benchmark allows us to point out some pros and cons of the NSC as a measure of problem difficulty for Evolutionary Algorithms. NSC is based on the concept of fitness cloud and works by partitioning the cloud into a number of bins representing as many different regions of the fitness landscape. The measure is calculated by joining the bins centroids by segments and summing all their negative slopes. Experiments presented in this paper show that if we fix a particular value for the minimum number of points that can be contained in a bin and if this number is, so to say, “not too large”, then the NSC seems to be able to correctly predict the problem difficulty of many *NK* landscape instances. In fact the NSC is more negative as the value of K increases and it is well known that *NK* landscapes difficulty also increases with K . Furthermore, we have been able to study some search spaces exhaustively and these results hold both when the space have been studied exhaustively and when it has been sampled by the Metropolis algorithm (thus confirming the suitability of this sampling technique for calculating the NSC, as it has already been shown in a number of contributions).

Nevertheless, the presented experiments also point out that the NSC tends to zero as the minimum number of points per bin increases. Thus, for too a large number of points per bin, the NSC fails to correctly pick up the difference in problem hardness of different *NK* landscapes instances.

All the above experimental results have been theoretically justified in this paper.

We conclude that, although the NSC is a useful hardness indicator, it has a number of drawbacks, principally bound to its many degrees of freedom, like the method used to partition fitness clouds into bins and the number of bins and number of points per bin. We believe that no general rule holds for obtaining these values. Thus, if the NSC can be calculated theoretically (as we have done for the *NK* landscapes), it is much more reliable to do so.

In the future, we plan to investigate more sophisticated methods for deciding the number of bins and the number of points per bin starting from the high level specifications of a problem, in order to obtain more precise NSC values. Finally, we plan to apply the NSC for quantifying the difficulty of real-life problems.

References

1. Aguirre, H.E., Tanaka, K.: Genetic algorithms on NK-landscapes: Effects of selection, drift, mutation, and recombination. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) *EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003*. LNCS, vol. 2611, pp. 131–142. Springer, Heidelberg (2003)

2. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Eshelman, L.J. (ed.) *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 184–192. Morgan Kaufmann, San Francisco (1995)
3. Kallel, L., Naudts, B., Rogers, A. (eds.): *Theoretical Aspects of Evolutionary Computing*. Springer, Heidelberg (2001)
4. Kauffman, S.A.: *The Origins of Order*. Oxford University Press, New York (1993)
5. Madras, N.: *Lectures on Monte Carlo Methods*. American Mathematical Society, Providence (2002)
6. Poli, R., Vanneschi, L.: Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. In: Thierens, D., et al. (eds.) *Genetic and Evolutionary Computation Conference, GECCO 2007*, pp. 1335–1342. ACM Press, New York (2007)
7. Rosé, H., Ebeling, W., Asselmeyer, T.: The density of states - a measure of the difficulty of optimisation problems. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) *PPSN 1996. LNCS*, vol. 1141, pp. 208–217. Springer, Heidelberg (1996)
8. Tomassini, M., Vanneschi, L.: Negative slope coefficient and the difficulty of random 3-sat instances. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008. LNCS*, vol. 4974, pp. 643–648. Springer, Heidelberg (2008)
9. Vanneschi, L.: Theory and Practice for Efficient Genetic Programming. Ph.D. thesis, Faculty of Science, University of Lausanne, Switzerland (2004),
<http://www.disco.unimib.it/vanneschi>
10. Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., Vérel, S.: Fitness clouds and problem hardness in genetic programming. In: Deb, K., et al. (eds.) *GECCO 2004. LNCS*, vol. 3103, pp. 690–701. Springer, Heidelberg (2004)
11. Vanneschi, L., Tomassini, M., Collard, P., Vérel, S.: Negative slope coefficient: A measure to characterize genetic programming fitness landscapes. In: Collet, P., et al. (eds.) *EuroGP 2006. LNCS*, vol. 3905, pp. 178–189. Springer, Heidelberg (2006)
12. Vérel, S., Collard, P., Clergue, M.: Where are bottleneck in NK fitness landscapes? In: CEC 2003: IEEE International Congress on Evolutionary Computation, Canberra, Australia, pp. 273–280. IEEE Press, Piscataway (2003)
13. Weinberger, E.D.: Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biol. Cybern.* 63, 325–336 (1990)

On the Parallel Speed-Up of Estimation of Multivariate Normal Algorithm and Evolution Strategies

Fabien Teytaud and Olivier Teytaud

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud), bat 490 Univ. Paris-Sud
91405 Orsay, France
fteytaud@lri.fr

Abstract. Motivated by parallel optimization, we experiment EDA-like adaptation-rules in the case of λ large. The rule we use, essentially based on estimation of multivariate normal algorithm, is (i) compliant with all families of distributions for which a density estimation algorithm exists (ii) simple (iii) parameter-free (iv) better than current rules in this framework of λ large. The speed-up as a function of λ is consistent with theoretical bounds.

1 Introduction

Evolution Strategies (ES [16]) are a robust optimization tool, known for its robustness (in particular in front of local minima) and simplicity. It is also known as suitable for parallel optimization, because it is population-based. However, it has been pointed out recently in [4] that usual step-size adaptation rules were far from being efficient for λ large, e.g. $\lambda = 4N^2$ where λ is the population size and N is the dimension of the search space.

The case of $\lambda = 4N^2$ as in [4] or $\lambda \gg 4N^2$ is not purely theoretical. In our recent applications, we usually optimized on a cluster of 368 cores, and we recently organized an optimization on several clusters of a few hundreds cores on each cluster. With $\lambda = 2000$ cores, $N = 22$ satisfies $\lambda = 4N^2$. Moreover, in many cases we have to launch several jobs simultaneously (more than the number of cores) in order to save up scheduling time, leading to λ much larger than the number of cores, and in robust optimization $N \leq 50$ is far from being trivial.

In this paper, we: (i) describe the main step-size adaptation rules in the literature (section 2); (ii) experiment step-size adaptation rules for various values of λ (section 3); (iii) discuss the results in section 4.

In all the paper, we assume that the reader is familiar with standard notations around ES (see e.g. [16,2] for more information) and we consider $(\mu/\mu, \lambda)$ -algorithms, i.e.: (i) at each iteration of the algorithm, λ points are generated according to some (usually but not necessarily) Gaussian distribution; (ii) the fitness values of these λ points are computed; (iii) then, the μ best points according to the fitness function are selected; averages are then w.r.t this subsample of the μ best points. All averages here are unweighted averages, but methods used

in this paper can be applied in weighted cases. $N(0, Id)$ and related notations (e.g. $N_i(0, Id)$) denote standard multivariate Gaussian random variables with identity covariance matrix.

2 Methods

A central issue in ES is the adaptation of the distribution (step-size and beyond). The one-fifth rule has been successfully introduced in [16] for this, and several other rules have been proposed later in the literature; some main rules are detailed in the rest of this section.

In this paper, we are considering minimization problems.

2.1 Cumulative Step-Size Adaptation (CSA)

Cumulative step-size adaptation has been proposed in [9]; essentially, this method compares the path followed by the algorithm to the path followed under random selection: if the followed path is larger, then the step size should increase. The detailed algorithm is presented in Algorithm 1.

Algorithm 1. Cumulative step-size adaptation.

```

Initialize  $\sigma \in \mathbb{R}$ ,  $y \in \mathbb{R}^N$ .
while halting criterion not fulfilled do
    for  $i = 1 \dots \lambda$  do
         $s_i = N_i(0, Id)$ 
         $y_i = y + \sigma s_i$ 
         $f_i = f(y_i)$ 
    end for
    Sort the individuals by increasing fitness;  $f_{(1)} < f_{(2)} < \dots < f_{(\lambda)}$ .
     $s^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} s_{(i)}$ 
     $y = y + \sigma s^{avg}$ 
     $p_{\sigma} = (1 - c)p_{\sigma} + \sqrt{\mu c(2 - c)}s^{avg}$ 
     $\sigma = \sigma \exp\left[\frac{\|p_{\sigma}\| - \bar{\chi}_N}{D\bar{\chi}_N}\right]$ 
end while

```

where $\bar{\chi}_N$ is $\sqrt{N} \times (1 - \frac{1}{4.0 \times N} + \frac{1}{21.0 \times N^2})$. Following [8], we choose $c = \frac{1}{\sqrt{N}}$ and $D = \sqrt{N}$. A main weakness of this formula is its moderate efficiency, for λ large, as pointed out in [4]. [4] therefore proposes mutative self-adaptation in order to improve the convergence rate as a function of λ ; this mutative self-adaptation is presented below.

2.2 Mutative Self-Adaptation (SA)

Mutative self-adaptation has been proposed in [16] and [19], and extended in [4] for improving the convergence rate for λ large. The algorithm is simple and provides the state of the art results for λ large; it is presented in Algorithm 2.

Algorithm 2. Mutative self-adaptation. τ is equal to $1/\sqrt{N}$; other variants have been tested (e.g. $1/\sqrt{2N}$ which is sometimes found in papers) without improvement in our case of λ large.

```

Initialize  $\sigma^{avg} \in \mathbb{R}$ ,  $y \in \mathbb{R}^N$ .
while Halting criterion not fulfilled do
  for  $i = 1.. \lambda$  do
     $\sigma_i = \sigma^{avg} e^{\tau N_i(0,1)}$ 
     $z_i = \sigma_i N_i(0, Id)$ 
     $y_i = y + z_i$ 
     $f_i = f(y_i)$ 
  end for
  Sort the individuals by increasing fitness;  $f_{(1)} < f_{(2)} < \dots < f_{(\lambda)}$ .
   $z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z_{(i)}$ 
   $\sigma^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} \sigma_{(i)}$ 
   $y = y + z^{avg}$ 
end while

```

2.3 Statistical Step-Size Adaptation (SSA)

We here propose simple step-size adaptation rules, inspired by Estimation of Distribution Algorithms (EDA), e.g. UMDA [14], Compact Genetic Algorithm

Algorithm 3. SSA. This is close to EMNA; we divide by $\mu \times N$ in the adaptation rule (eq. defining σ) because the sum is over $\mu \times N$ deviations (one per selected individual and per axis), and we add a trick against premature convergence. The constant K is here ∞ as this work is not devoted to premature convergence; however, we verified that the same convergence rates as those presented in this paper can be recovered with this more robust version.

```

Initialize  $\sigma \in \mathbb{R}$ ,  $y \in \mathbb{R}^N$ .
while Halting criterion not fulfilled do
  for  $l = 1.. \lambda$  do
     $z_l = \sigma N_l(0, Id)$ 
     $y_l = y + z_l$ 
     $f_l = f(y_l)$ 
  end for
  Sort the individuals by increasing fitness;  $f_{(1)} < f_{(2)} < \dots < f_{(\lambda)}$ .
   $z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z_{(i)}$ 
  if  $\|z^{avg}\| < K\sigma$  then
     $\sigma = \sqrt{\frac{\sum_{i=1}^{\mu} \|z_{(i)} - z^{avg}\|^2}{\mu \times N}}$ 
  else
     $\sigma = 2\sigma$  /* this avoids premature convergence in case of  $\sigma$  poorly chosen*/
  end if
   $y = y + z^{avg}$ 
end while

```

[10], Population-Based Incremental Learning [1], Relative Entropy [13], Cross-Entropy [5] and Estimation of Multivariate Normal Algorithms (EMNA) [11] (our main inspiration), which combine (i) the current distribution (possibly), (ii) statistical properties of selected points, into a new distribution. We show in this paper that forgetting the old estimate and only using the new points is a good idea in the case of λ large; in particular, premature convergence as pointed out in [20,7,12,15] does not occur if $\lambda \gg 1$ points are distributed on the search space with non-degenerated variance, and troubles around variance estimates for small sample size as in [6] are by definition not relevant for us. Its advantages are as follows for λ large: (i) it's very simple and parameter free; the reduced number of parameters is an advantage of mutative self adaptation in front of cumulative step-size adaptation, but we show here that yet fewer parameters (!) is possible and leads to better results, thanks to EMNA-like algorithms; (ii) we use it here in the case of a Gaussian, but it could easily be used also for any density estimation technique e.g. sums of Gaussians (i.e. no problem with multimodal distributions); (iii) it could easily be used for discrete spaces also. The algorithm is presented in Algorithm 3 for estimating only one step-size, but a straightforward extension is one step-size per axis or even a full covariance matrix (see section 3.3).

3 Experimental Results

As we here focus on step-size adaptation-rules, we here do not consider complete Covariance-Matrix Adaptation (CMA) but only diagonal covariance matrix adaptation, but [17] has shown that this diagonal framework is indeed better in many (even non separable) cases, and we point out that all methods discussed in section 2 have anyway straightforward extensions in the framework of complete covariance matrix or covariance matrix by block. We first confirm results from [4] (superiority of mutative self-adaptation over cumulative step-size adaptation for λ large), in section 3.1; we then validate the statistical step-size adaptation rule in section 3.2. Table 1 presents the objective functions considered in this paper.

Table 1. Objective functions considered in this paper

Name	Objective function
Sphere	$f(y) = \sum_{i=1}^N y_i^2$
Schwefel	$f(y) = \sum_{i=1}^N (\sum_{j=1}^i y_j)^2$
Cigar	$f(y) = y_1^2 + 10000 \times \sum_{i=2}^N y_i^2$

3.1 Validating SA for λ Large

We here confirm results in [4], namely the moderate efficiency of CSA for λ large (at least under the various parameterizations we tested). Following [4], Figure 1 presents the numbers of iterations before a fixed halting criterion ($f_{stop} = 10^{-10}$).

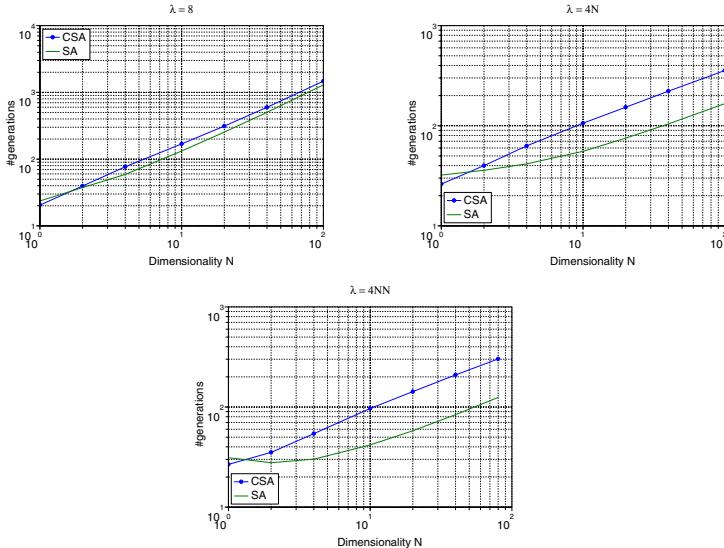


Fig. 1. Performance of CSA and SA on the sphere function (number of iterations before $f(x) < 10^{-10}$) depending on the dimensionality. We confirm here the superiority of SA on cumulative step-size adaptation, at least for our version of SA and CSA. In these experiments, $\mu = \frac{1}{4}\lambda$.

3.2 Validating the Statistical Step-Size Adaptation

We now present the main result of this paper, i.e. the efficiency of the statistical step-size adaptation rule. Results are presented in Fig. 2, 3, 4 for $\mu = \lambda/2$, $\mu = \lambda/4$, $\mu = 1$ respectively. Presented curves are $N \times \log(||x - x^*||)/n$ as a function of λ , where: N is the dimensionality; x is the best point in the last λ offspring; x^* is the optimum ($x^* = 0$ for our test functions); n is the number of iterations before the halting criterion is met. Each run is performed until $f(x) < 10e^{-50}$. For the sake of statistical significance each point is the average of 300 independent runs.

3.3 Anisotropic Case

The SSA algorithm can be adapted to the anisotropic case. We here consider the separable version, i.e. a diagonal covariance matrix for the Gaussian; this form of covariance matrix is intermediate between the full matrix and the matrix proportional to identity. It has been pointed out in [17] that this separable

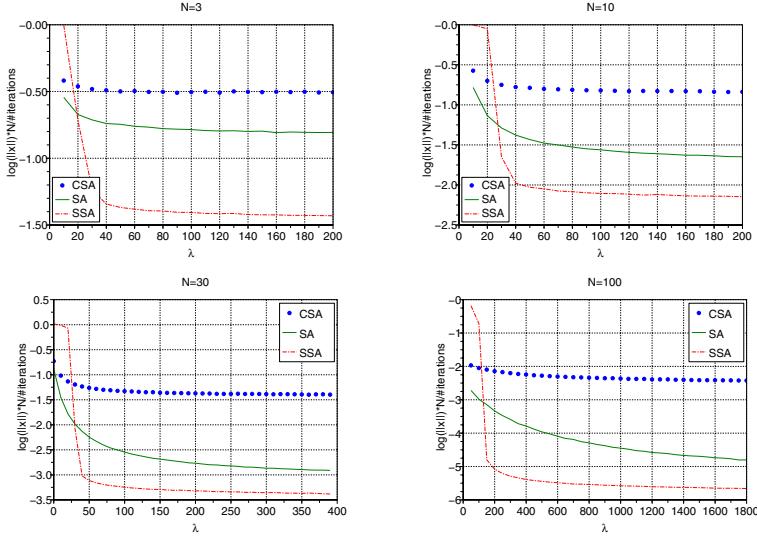


Fig. 2. Log of distance to the optimum when the halting criterion (see text) is met, normalized by the dimension and the number of iterations. Results for $\mu = \lambda/2$, on the sphere function. In all cases, SSA is the best rule for λ large.

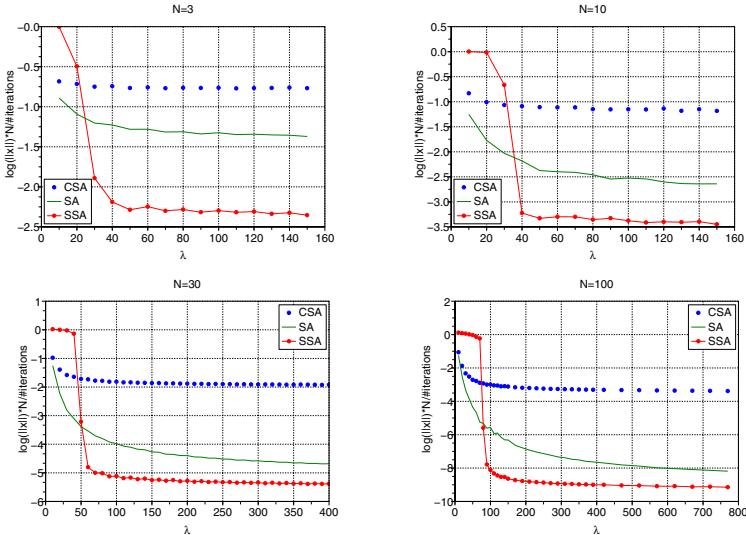


Fig. 3. Results for $\mu = \lambda/4$ on the sphere function. All methods are better than for $\mu = \lambda/2$. In all cases, SSA is the best rule for λ large.

version is in many cases faster than the complete covariance matrix. We have in this case to determine one step-size per axis; see Algorithm 4.

Figure 5 presents the results of Algorithm 4. We can see that: (i) On the Schwefel function, the results are the same as in the case of the sphere function.

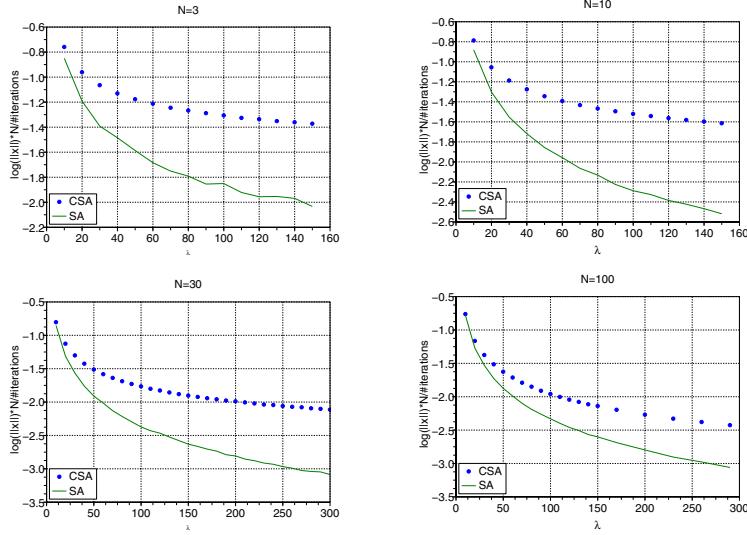


Fig. 4. Results for $\mu = 1$ on the sphere function. SSA is not presented as it does not make sense for $\mu = 2$. As shown by this figure (compared to Figs 3 and 2), $\mu = 1$ is quite weak for large dimension and the absence of SSA version for that case is therefore not a trouble.

Algorithm 4. Anisotropic statistical step-size adaptation. We divide the average squared deviation by μ in the step-size adaptation rule (equation defining σ), instead of $\mu \times N$ in the anisotropic case, because now the step-size is averaged over μ squared deviations only (one per selected individual). The subscript j refers to the axis.

Initialize $\sigma \in \mathbb{R}^N$, $y \in \mathbb{R}^N$.

while Halting criterion not fulfilled do

for $l = 1.. \lambda$ do

for $i \in \{1, \dots, N\}$ do

$$z_{l,i} = \sigma_j N_{l,j}(0, 1)$$

end for

$$y_l = y + z_l$$

$$f_l = f(y_l)$$

end for

Sort the individuals by increasing fitness; $f_{(1)} < f_{(2)} < \dots < f_{(\lambda)}$.

$$z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z_{(i)}$$

for $j \in \{1, \dots, N\}$ do

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{\mu} \|z_{(i),j} - z_j^{avg}\|^2}{\mu}}$$

end for

$$y = y + z^{avg}$$

end while

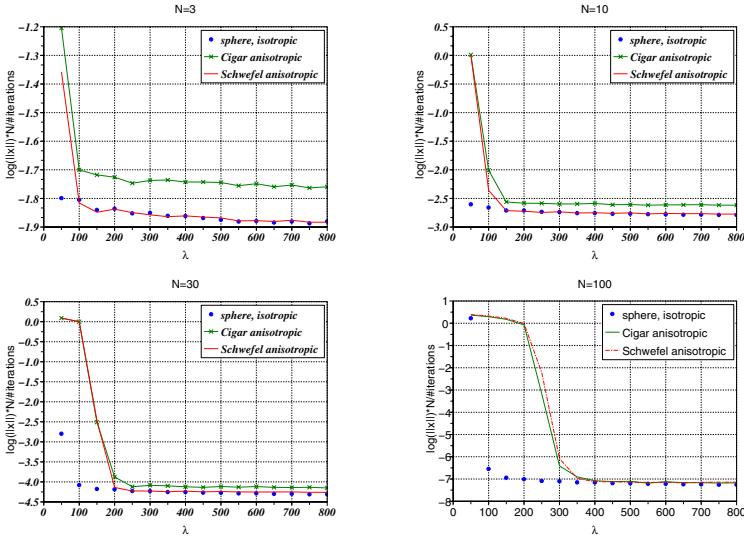


Fig. 5. We here present the results of the anisotropic version of the SSA algorithm (Algorithm 4). It has on ill-conditioned functions nearly the same performance as the isotropic version on the sphere, in particular in high dimension.

The isotropic algorithms have, in this framework (non presented results), a result close to 0 (i.e. much worse). Therefore, the anisotropic step-size adaptation works for this moderately ill-conditioned function. (ii) On the Cigar function, the results are not exactly those of the sphere function, but almost; even on this much more ill-conditioned function, the anisotropic SSA works.

4 Discussion

First, we confirm the superiority of mutative self-adaptation over cumulative step-length adaptation, as already shown in [4], in the case of λ large. However, possibly, tuning CSA might lead to improved results for high values of λ ; this point, beyond the scope of this paper, is not further analyzed here. Second, we experiment a simple statistical adaptation rule adapted from EMNA. This rule for step-size adaptation is (i) very simple (the most simple of all rules in section 2) (ii) fully portable to the full-covariance case (as well as the SA approach used in [4] for covariance matrix adaptation) (iii) computationally cheap (iv) highly intuitive (v) parameter-free (except K if we include the rule against premature convergence, which is only required if choosing σ sufficiently large is hard) (vi) efficient whenever λ is not known in advance as λ has no impact on the adaptation (important for fault-tolerance). It provides a speed-up of 100% (over mutative self-adaptation, which is already much better than CSA) on the sphere function for $N = 3$, decreasing to 33% for $N = 10$ and 25% for $N = 100$ (in the case $\lambda = 150$). The usual adaptation-rules use a combination

of an old covariance matrix and a new estimate based on the current population - essentially, in SSA, we only keep the second term as λ large simplifies the problem. We point out that we only experimented the Gaussian case so that we can compare our results to standard rules, but our algorithm has the advantage that it can be adapted to *all* distributions for which the parameters can be estimated from a sample. Sums of Gaussians are a natural candidate for further work. Third, we show that the anisotropic version works in the sense that the convergence rate on the sphere was recovered with the Schwefel and the Cigar function.

Some by-products of our results are now pointed out, as a comparison with theoretical results in [21]: (i) The higher the dimensionality, the better the speed-up for $(\mu/\mu, \lambda)$ -algorithms; [3] has shown the linear speed-up as a function of λ as long as $\lambda \ll N$, and [21] has precised formally that the speed-up is linear until λ of the same order as the dimension. Roughly, a number of processors linear as a function of the dimension is efficient. This is visible on our experimental results. (ii) Also, $(1, \lambda)$ algorithms (case $\mu = 1$) have a less-than-linear (logarithmic) speed-up as a function of λ . This is visible in our experimental results. This is also consistent with [3] and [21]. (iii) We have proposed a very simple rule and got state-of-the-art results. This suggests that there is much to win by a refined work on the important case of λ large.

Acknowledgements. We are grateful to H.-G. Beyer for useful email answers, and to A. Auger, N. Hansen, R. Ros for fruitful discussions. We also thank the MoGo-people (G. Chaslot, J.-B. Hock, A. Rimmel) for interesting discussions around parallel applications of ES.

References

1. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning, Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (1994)
2. Beyer, H.-G.: The Theory of Evolution Strategies. Natural Computing Series. Springer, Heidelberg (2001)
3. Beyer, H.-G.: The Theory of Evolutions Strategies. Springer, Heidelberg (2001)
4. Beyer, H.-G., Sendhoff, B.: Covariance matrix adaptation revisited - the CMSA evolution strategy. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 123–132. Springer, Heidelberg (2008)
5. Cai, Y., Sun, X., Xu, H., Jia, P.: Cross entropy and adaptive variance scaling in continuous eda. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 609–616. ACM Press, New York (2007)
6. Donga, W., Yao, X.: Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms. Information Sciences 178(15), 3000–3023 (2008)
7. Grahl, J., Bosman, P.A., Rothlauf, F.: The correlation-triggered adaptive variance scaling idea. In: GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 397–404. ACM, New York (2006)

8. Hansen, N.: Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie. Eine Untersuchung zur entstochastisierten, koordinatenistemus-nahmigen Adaptation der Mutationsverteilung. Mensch und Buch Verlag, Berlin (1998) ISBN 3-9333346-29-0
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
10. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. *IEEE Trans. on Evolutionary Computation* 3(4), 287 (1999)
11. Larranaga, P., Lozano, J.A.: *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Dordrecht (2001)
12. Liu, J., Teng, H.-F.: Model learning and variance control in continuous edas using pca. In: ICICIC 2008: Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control, Washington, DC, USA, p. 555. IEEE Computer Society, Los Alamitos (2008)
13. Mühlenbein, H., Höns, R.: The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation* 13(1), 1–27 (2005)
14. Mühlenbein, H., Mahnig, T.: Evolutionary computation and Wright's equation. *Theoretical Computer Science* 287(1), 145–165 (2002)
15. Posík, P.: Preventing premature convergence in a simple eda via global step size setting. In: Rudolph, et al. (ed.) [18], pp. 549–558
16. Rechenberg, I.: *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart (1973)
17. Ros, R., Hansen, N.: A simple modification in cma-es achieving linear time and space complexity. In: Rudolph, et al. (ed.) [18], pp. 296–305
18. Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.): *PPSN 2008. LNCS*, vol. 5199. Springer, Heidelberg (2008)
19. Schwefel, H.-P.: Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re 215/3, Technische Universität Berlin (July 1974)
20. Shapiro, J.L.: Drift and scaling in estimation of distribution algorithms. *Evolutionary Computation* 13(1) (2005)
21. Teytaud, O., Fournier, H.: Lower bounds for evolution strategies using vc-dimension. In: Rudolph, et al. (ed.) [18], pp. 102–111.

Adaptability of Algorithms for Real-Valued Optimization

Mike Preuss

Technische Universität Dortmund, Chair of Algorithm Engineering
44221 Dortmund, Germany
mike.preuss@tu-dortmund.de

Abstract. We investigate the adaptability of optimization algorithms for the real-valued case to concrete problems via tuning. However, the focus is not primarily on performance, but on the tuning potential of each algorithm/problem system, for which we define the *empirical tuning potential* measure (ETP). It is tested if this measure fulfills some trivial conditions for usability, which it does. We also compare the best obtained configurations of 4 adaptable algorithms (2 evolutionary, 2 classic) with classic algorithms under default settings. The overall outcome is quite mixed: Sometimes adapting algorithms is highly profitable, but some problems are already solved to optimality by classic methods.

1 Introduction

The *evolutionary computation* (EC) field is on the move. After years of self-containedness, the necessity to demonstrate the effectiveness of *evolutionary algorithms* (EAs) in direct comparison to other optimization algorithms which are frequently applied for solving engineering problems has become obvious. However, such studies do already exist, as e.g. provided by Schwefel already in 1979 [14]. Since then, the usage of many algorithms or benchmark problem sets has almost remained the same. However, the enormously increased computing power entailed a matching growth in the availability of experimental results, which lead to some new insight.

Triggered by several warning voices who critisized the arbitrariness of published EC algorithm comparison studies (one of which belongs to Eiben and Jelasity [9]), researchers of the field have put some effort into strengthening the experimental methodology, e.g. by standardization of test problems and performance measures. Additionally, the enormous effects of algorithm parametrization have been unveiled, and consequently, tuning methods were presented, like the *sequential parameter optimization* (SPO) [2], or the F-race by Birattari et al. [6], the latter being especially well suited for combinatorial optimization. Other approaches like meta-EAs tackle the same goal. These methods adapt the parameters of an algorithm to a concrete setting which shall contain a single problem or small problem class, the desired performance measure, and other environmental conditions as a maximum run length. On a first glance, the best achieved performances are of great value, because they let us compare algorithms under fair conditions, approximately at the top of their capacity.

Provided with automated tuning methods, we may also tackle the question for the *adaptability* of optimization algorithms towards a given situation, with the overall task to generalize on this algorithm property. Adaptability does not only consider top performance, but also the relative gain when starting from an ‘average’ (default?) algorithm configuration, as well as the effort needed to get there. To give an example, algorithm A may perform reasonably well on some problems, but does not allow for much adaptation because it does not provide any parameters which can be employed as handles for modifying the algorithm. Algorithm B may perform worse under default parameter values, but enables changing its behavior so as to obtain much better results by means of a tuning process. Initially, there is good reason to prefer algorithm A, but if tuning can be applied, algorithm B may turn out to be much better due to its adaptability.

Of course, an optimization algorithm may also be improved by introducing new or modified operators reflecting problem knowledge. However, there is little chance to do that in an automated and thereby measurable way, whereas tuning methods may be applied as soon as some parameters are available which can be adapted. Yet, it is hard to measure adaptability without imposing too many constraints on the employed problems. It shall be possible even in situations when the global optimum (of the optimization problem) is not known. It should also not depend on detecting optimal parameters for an algorithm, as this cannot be guaranteed by the tuning methods. We need to define a measure and then do an empirical investigation to see if the definition proves worthwhile. This work provides a first attempt into that direction and obeys the following scheme: §2 concretizes the aims pursued and introduces the chosen test problems and optimization algorithms. §3 discusses the meaning of parameters and suggests an adaptability measure. §4 reports on a thorough experimental investigation, followed by conclusions.

2 Aims and Methods

We want to investigate two related aspects of adaptability:

- How may adaptability be measured, and what do the measures tell us, especially concerning ‘classic’ (e.g. quasi-Newton) optimization algorithms?
- For any single algorithm, does it really pay off to tune? Is there a significant difference between default and tuned configurations’ performances?

For automated parameter tuning, we employ the *sequential parameter optimization* (SPO) [2] algorithm, in the variant suggested in [3]. Starting from a *latin hypercube sample* (LHS) based kriging model, it uses random permutation tests to decide if a suggested new configuration is better than an old one. The maximum budget is always 1000 algorithm runs, with a minimum of 4 repeats. For more than 2 parameters, a grid search would be much more expensive. The tuning results are investigated using Kendall’s non-parametric tau test for correlations; random permutation tests are employed for detecting significant differences between different algorithm configurations.

Table 1. Selected problems from the CEC 2005 library and some of their properties. No problem is separable and/or unimodal. All problems used in 10 dimensions.

no	problem	multimodal characteristic	optimum	domain
6	shifted Rosenbrock's	yes narrow valley from local to global optimum	390	-100/100
10	shifted rotated Rastrigin's	highly regular local optima structure	-330	-5/5
12	Schwefel's 2.13	yes best optima are far from each other	-460	$-\pi/\pi$
13	(F8F2) shifted expanded Griewank's plus Rosenbrock's	extremely global structure very flat	-130	-5/5
16	rotated hybrid composition 1	extremely irregular with local patterns and plateaus	120	-5/5
17	rotated hybrid composition 1 with fitness noise	extremely same as 16 plus noise	120	-5/5
23	non-continuous rotated hybrid composition 3	highly local patterns with many plateaus	360	-5/5

2.1 Test Problems

We select some problem instances from the CEC 2005 contest library [15], as enlisted in table 1. These may all be considered as hard, with different degrees of multimodality, and some additional properties like non-continuity and added noise. None of the problems is separable. This choice is motivated by the hope to discriminate the performance of the algorithms according to different problem characteristics. We acknowledge that the chosen instances are generally not very well suited for the classic optimization methods like the one of *Broyden-Fletcher-Goldfarb-Shanno* (BFGS). However, by allowing multistarts, in principle every method can reach the global optimum, and the CEC 2005 contest was intended to be hard, as simple and smooth problems are the typical domain of gradient-based optimization algorithms.

2.2 Algorithms

For the classical optimization algorithms, we basically rely on the *optim* method in the R stats package, which provides a quasi-Newton BFGS, a *conjugate gradients* (CG), and a Nelder-Mead [13] method. These three are derived from the implementation of Nash [12]. Furthermore, we employ the L-BFGS-B algorithm of Byrd et al. [7] and the *simulated annealing* (SA) variant of Belisle [4] contained in optim. Additionally, the alternative BFGS (hereafter denoted ALT-BFGS) implementation of Clausen [8] is used, as it provides access to several parameters usually hidden in backend libraries.

From the domain of metaheuristics, we test two related evolutionary algorithms, namely a generic *evolution strategy* (ES) with self adaptation as suggested by Schwefel [5], and the *covariance matrix adaptation evolution strategy* (CMA-ES) by Hansen and Ostermeier [10].

3 Parameters and Adaptability

What does it conceptually mean to adapt parameters of optimization algorithms? We shall question an often voiced opinion which marks large parameter

sets as bad and strives for reducing their size. So what are the advantages and disadvantages of parameterized (optimization) algorithms? On the negative side, we have at least two:

- A large set of parameters makes a method more difficult to handle, because at least the unexperienced user does not know how to set these right.
- Parameter-parameter and parameter-problem interactions may enormously complicate evaluating algorithm performance, also making it more difficult to obtain good parameter settings.

Nevertheless, we here want to further a slightly unorthodox view of parameters, namely as handles to modify algorithms as desired. Simple standard algorithms are usually considered as being parameterless, but may easily be extended into parameterized ones (e.g. clever-quicksort). However, it is the advantage of simple algorithms with only one parameter that one can often prove a particular parameter value to deliver optimal performance. For evolutionary algorithms running on different optimization problems, this approach is not feasible. This is partly due to their rather complex stochastic behavior, but also stems from the inherent imprecision of the term *algorithm* as it is often applied to EAs. A concrete algorithm is instantiated only if the two following conditions are met.

1. The problem is clearly specified (not e.g., an ES applied to TSP).
2. All (exogenous) parameter values are fixed.

If this is not the case, one actually deals with an algorithm family, and one usually does so based on theoretical or empirical findings obtained from single algorithm instances. Whereas the first condition leads us to an uninevitable dilemma between practical usability and generalizability and thus between the real-world application and the scientific approach, the second one can be dealt with e.g. by parameter tuning. But still, it is not obvious which of the resulting tuned instances a scientific inquiry shall be based on. The best that is obtained by a concrete tuning method with a given time limit? Or a reasonably good (e.g. average) one? De Jong [11] points out that ‘getting in the ball park’ by achieving a reasonably well performing parameter setting is often sufficient for practical purposes, and that doing so may not be too difficult because of a certain robustness of EAs towards parameter changes. However, even for classic optimization algorithms for the real-valued domain, adapting parameters to the concrete problem may be advantageous. This is probably so for highly multimodal or otherwise considered difficult problems. Nevertheless, adaptation comes at a cost, and one shall take into account how difficult it is to ‘get into the ball park’, as well as the achieved performance differences relative to default parameter configurations.

3.1 Adaptability

When treating a real-world problem, (evolutionary) optimization algorithms are often adapted after a simple scheme. Based on the experience of the algorithm designer, a canonical EA is adjusted according to the interpretation of first results, thereby largely following intuitive reasoning. However, no generally

applicable structured method exists to attain a good optimization algorithm for a yet untreated problem. Besides, evolutionary algorithms are so flexible that it seems unreasonable to approach such a method. Parameter tuning may help to a certain extent, but only insofar as a) there is something to tune, that is, the employed operators are parametrized, and b) the chosen components of the tuned algorithm are well suited to the problem at hand.

However, availability of fairly automated tuning procedures enables us to look at the adaptation process from the opposite direction: *What does parameter tuning of an optimization algorithm towards one or several problems tell us about the algorithm?* Applying a tuning method lets us obtain an estimate for peak and average performance. It surely depends on the employed performance measure how these two values can be aggregated into one, but this quantity may then serve as a measure for the tuning potential of an algorithm-problem system. However, we also need to make the assumption that the peak performance detected via tuning is a good estimator of the achievable maximum. The accuracy of this claim clearly depends on the quality of the employed tuning method, and its proper use. Time may be an important factor that limits tuning quality, because tuning requires multiple optimization algorithm runs, which is very costly for a non-trivial problem. Leaving these objections aside, and presuming that we have obtained two samples representing peak \mathbf{y}_p and average performance \mathbf{y}_a , we suggest to compute an *empirical tuning potential* (ETP) measure (for minimization) from these samples. We deliberately abstain from using any ‘target’ performance as it is not generally known.

$$\text{ETP}(\mathbf{y}_p, \mathbf{y}_a) := \frac{\text{median}(\mathbf{y}_a) - \text{median}(\mathbf{y}_p)}{\text{sq}(\mathbf{y}_a)} \cdot \frac{\text{median}(\mathbf{y}_a) - \text{median}(\mathbf{y}_p)}{\text{sq}(\mathbf{y}_p)} \quad (1)$$

In the given formula, sq stands for semi-quartile range, meaning half of the distance between the lower and the upper quartile. Its advantage lies in the fact that it allows for an estimation of the spread without depending on a specific distribution type (e.g. normal). The ETP consists of two components, namely the relative improvement and the spreads. The rationale behind the latter is that algorithms which mostly reach a near optimal value usually exhibit a much smaller spread than ones that often get stuck far away from it. We thus employ distribution properties of the average and the best performance sample to describe how good the improved value is.

4 Experimental Investigation

When optimizing a real-world problem, time is usually the most important constraint. But as one often possesses a good estimation of the expected time for each function evaluation, one can easily give the maximum number of evaluations allowed. The concrete value of this number may be specific to every single optimization task, but according to our experience, $1E3$ to $1E4$ is a reasonable size for many applications. We therefore run each algorithm twice, with a budget of $1E3$ and $1E4$ evaluations.

Table 2. Median (51 runs) performance of classic algorithms under default settings

	ALT-BFGS		BFGS		CG		L-BFGS-B		SANN		Nelder-Mead	
no.	1000	10000	1000	10000	1000	10000	1000	10000	1000	10000	1000	10000
6	3.82E9	393.98	390.00	390.00	390.00	390.00	390.00	390.00	1.48E10	399.77	8.77E5	3.67E4
10	-173.30	-260.35	-144.94	-254.88	-133.99	-216.57	-143.44	-235.97	-178.89	-192.90	-23.087	-113.05
12	3.18E5	-296.21	-459.99	-459.99	-459.99	-459.99	-459.99	-459.99	1670.8	-113.98	2498.9	-274.72
13	81.314	-112.10	-123.64	-128.01	-122.34	-124.68	-83.996	-113.10	-124.78	-126.65	-99.094	-126.16
16	1144.4	587.90	638.97	477.68	794.33	465.56	809.03	521.49	735.76	636.86	704.18	474.07
17	1124.2	745.19	1064.2	621.97	1051.3	621.25	1624.9	1070.5	871.62	660.13	1660.7	1097.5
23	1953.9	1816.1	1948.2	1774.1	2052.3	1785.8	2112.6	1908.6	1658.4	1661.3	1986.7	1724.3

Whereas most classic optimization algorithms as BFGS and CG have internal mechanisms to detect when to stop, this is not necessarily true for metaheuristics. As they often do not possess explicit gradient or hessian information of their current search region (the CMA-ES is a notable exception here), defining such a criterion is not trivial. For the generic ES, we thus employ a very simple heuristic and terminates the search if no progress has been made after 10 generations.

In order to enable a fair comparison and use up the maximum number of function evaluations, a simple restart mechanism is applied. It starts the algorithm anew at a random location whenever the budget is not yet exceeded. For all algorithms which terminate autonomously and do not give us access to the code that does so (true for all methods of R optim), we estimate the expected number of function evaluations needed for each single search by averaging the finished searches. Whenever the number of consumed evaluations plus the expected number of evaluations for one search lies below `max evaluations +10%`, the algorithm is given another random start. This mode of operation adds some variation to the results, as the algorithms do not always use *exactly* $1E3$ or $1E4$ evaluations, but some number near to it. However, we expect that this influence is rather small for $1E4$, where often many restarts are done, and unfortunately unavoidable for $1E3$, as long as the algorithms cannot be told from the start how many evaluations to use. As all tested problems are multimodal, it is clear that we cannot expect a single run to detect the global optimum in all cases as it may be trapped in a local one.

Experiment: How adaptable are the tested optimization algorithms via tuning?

Pre-experimental planning. As a reference for comparison with the adaptable algorithms, a performance table of the considered classical methods is derived, depicted in table 2. All algorithms are run 51 times on each problem and with a budget of $1E3$ and $1E4$ evaluations, respectively. The reported median values indicate that most algorithms cope relatively well with the test problems, with the exception of the Nelder-Mead algorithm, which performs considerably worse. The ALT-BFGS method obviously needs more than $1E3$ evaluations to reach good areas on problems 6 and 12, but with $1E4$ evaluations it also performs well.

According to some preliminary tests, we fixed the parameter intervals for the adaptable methods, given in table 3. Especially for the C_{cov} parameter of the CMA-ES, it was found that values of more than 1 usually led to failure of the algorithm, thus the chosen interval was set to the range from 0 to 1. The restart base parameter of the CMA-ES is a generalization of the restart with

increasing population size as introduced in [1]. Each time a restart is scheduled, the population size is not just doubled, but multiplied with this factor. The $X \cdot C_s$ and $X \cdot C_c$ parameters are simply multipliers for the internally precomputed original C_s and C_c values. For the generic EA, we introduce the p_{reco} parameter as probability of recombination. As the treated problems are all multimodal, employing mutation only to realize independent searches may make sense.

Task. We cannot demand a concrete adaptability value (ETP) for any algorithm as there is no comparison data on these values available. However, we shall expect that a) if and only if the median and tuned configuration are significantly different (5% random permutation test), than the ETP should have a large value (> 1), and b) that there is a negative rank correlation between the p-values of the significance test between median and tuned, and the ETP, when cross-tabled. We require a correlation of at most around -0.5 to state that the measure makes sense. Concerning the reviewed adaptable algorithms, the task is to beat the classic methods by changing parameter configurations according to the problem. So at least one of the adatable methods shall be significantly better than the best non-adaptable algorithm (see table 2), detected by a random permutation test at the 5% level. Problems which are solved optimally by the non-adaptable algorithms are excluded here.

Setup. For each of the four adaptable algorithms (Nelder-Mead, ALT-BFGS, CMA-ES, and generic-ES) and each run-length (1E3 and 1E4), an SPO run is executed with a budget of 1000 runs, starting from an LHS of size 100 with 4 runs per algorithm. The allowed parameter intervals are given in table 3. By means of eq. 1, we compute the ETP from the median configuration of the LHS and the best configuration obtained via SPO, each validated with 51 separate runs. Additionally, a random permutation test is performed between these two, to check if there is a significant difference in quality stemming from the adaptation (tuning) process. We shall state that for the CMA-ES as well as for the generic-ES, default parameter guidelines are known, and it surely makes sense also to compare with these (resulting in a different ETP definition which we defer to future work). However, here we deliberately take possible misconfigurations into account, as may happen within allowed parameter bounds. Experience shows that for most algorithms and suitable bounds, the median configuration of a large LHS already performs quite well. Algorithms that suffer dramatically from every deviation from the default/best parameters are penalized by this measure.

Table 3. Parameter intervals for the tuning process

Nelder-Mead	alpha	beta	gamma				
	0:1:5	0:1:5	0:1:5				
ALT-BFGS	initial stepsize	ftol	gtol				
	domain:(0.002:1)	-6:-2	0.1:0.99				
self-adaptive EA	initial stepsize	population	selection pressure τ	p_{reco}			
	domain:(0.002:1)	2:100	2:10	0.01:1	0.01:1		
CMA-ES	initial stepsize	population damp	multiplier $X \cdot C_s$	$X \cdot C_c$	C_{cov}	restart	base
	domain:(0.002:1)	2:100	0.2:5	0.2:2	0.2:2	0:1	1:5

Table 4. CMA-ES: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

no.	1000					10000					
	LHS	median	SPO	best	p-value	ETP	LHS	median	SPO	best	p-value
6		8.24E7	804.06	1e-04	17731			397.38	390.00	0.0189	110.64
10		-270.44	-313.09	1e-04	100.24			-325.03	-328.01	1e-04	5.9983
12		25386	6506.3	1e-04	8.4837			1096.7	-449.99	1e-04	1.0539
13		-118.74	-127.41	1e-04	52.423			-128.97	-129.22	1e-04	1.3046
16		440.26	277.85	0.2499	13.613			229.30	213.45	1e-04	10.009
17		502.22	335.70	1e-04	23.990			250.80	219.94	1e-04	30.324
23		1601.0	1566.6	0.0216	0.0801			1330.5	919.56	1e-04	2.6853

Table 5. Generic-ES: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

no.	1000					10000					
	LHS	median	SPO	best	p-value	ETP	LHS	median	SPO	best	p-value
6		8.86E9	3951.2	1e-04	1.00E6			1537.4	483.59	1e-04	14.870
10		-206.99	-309.82	1e-04	156.75			-310.60	-320.05	1e-04	19.240
12		93010	6891.3	1e-04	76.331			163.58	-313.40	4e-04	1.5113
13		4526.5	-126.97	1e-04	3598.5			-128.15	-129.47	1e-04	14.928
16		2572.7	1667.4	1	9.1604			1023.6	426.24	1e-04	32.024
17		2652.4	1887.9	1	4.6060			1344.1	482.69	1e-04	41.054
23		2980.4	1931.5	1e-04	22.579			1703.4	1440.2	1e-04	5.4864

Table 6. Nelder-Mead: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

no.	1000					10000					
	LHS	median	SPO	best	p-value	ETP	LHS	median	SPO	best	p-value
6		3.22E8	9547.1	1e-04	3189.8			9.00E8	1078.6	1e-04	9.39E6
10		-195.60	-205.68	0.0834	0.5810			-232.89	-235.91	0.0548	0.1646
12		53157	-23.822	1e-04	270.61			26886	-459.97	1e-04	3.99E7
13		542.85	-125.56	1e-04	320.52			-38.399	-128.54	1e-04	334.50
16		591.41	549.66	5e-04	1.6421			477.60	400.94	1e-04	5.8551
17		668.74	662.37	0.0524	0.0118			502.12	495.95	0.0972	0.0278
23		1776.8	1678.9	1e-04	10.943			1692.0	1621.0	1e-04	15.104

Results/Visualization. The results of the tuning process are depicted together with the computed ETP in tables 4, 5, 6, and 7. Kendall's tau test between the attained ETP values and the corresponding p-values discriminating between peak and median configuration from the four tables gives a correlation estimate of -0.554 with a p-value of 5.2E-8. Note that the discriminating power of the ETP is higher than the p-value as it returns intermediate values where the p-value is either near zero or 1, see e.g. table 5. Concerning the comparison of the best unparametrized method with the best adaptable method, we state that there is no need for a better method for problems 6 and 12 (1E3 and 1E4 evaluations); these are solved to optimality by BFGS, CG, and L-BFGS-B. On problem 10, we have to compare SANN against the CMA-ES (1E3) and ALT-BFGS against the CMA-ES (1E4). For problem 13, it is SANN vs. CMA-ES (1E3) and BFGS vs. the generic ES (1E4). Problem 16 is dominated by BFGS and the CMA-ES (1E3), and CG and the CMA-ES (1E4). On problem 17, SANN and the CMA-ES perform best (1E3), and CG and the CMA-ES (1E4). Finally, for problem

Table 7. ALT-BFGS: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

no.	1000					10000					
	LHS	median	SPO	best	p-value	ETP	LHS	median	SPO	best	p-value
6	4.44E10	4.82E7	1e-04	153.11			8.80E8	390.00	1e-04	9.52E15	
10	-162.85	-175.78	0.1385	0.1093			-247.41	-259.35	0.1624	0.6133	
12	3.30E6	1233.6	1e-04	27.228			-163.85	-459.99	1e-04	1.22E11	
13	136.59	32.148	0.3409	0.3152			-102.06	-122.95	0.0019	0.5457	
16	1026.6	1090.2	0.7604	0.0931			610.37	626.33	0.6361	0.0377	
17	1110.5	1090.2	0.5835	0.0142			762.43	708.35	0.1577	0.5200	
23	1911.0	1921.6	0.3728	0.0291			1810.8	1813.7	0.4411	0.0066	

23, we compare SANN against the CMA-ES ($1E3$ and $1E4$). In all cases, the adaptable algorithm was significantly better than the unparametrized one, with random permuation test p-values of below $1.0E - 4$. Only on problem 23 ($1E3$), the p-value was larger (0.0042), still indicating a significant difference.

Observations. We observe that for most algorithms, tuning works out well, so that the performance increases significantly. However, the ALT-BFGS seems to be hard to tune as indicated by best configurations in the range of median configurations. It is also striking that SANN performs remarkably well, especially on the more rugged landscapes on problems 13, 17, and 23. However, for each algorithm, the measured tuning potential is quite different for the considered problems: E.g., for Nelder-Mead, there is almost no potential on problems 10 and 17, but there is for the others. The two evolutionary algorithms (CMA-ES and generic ES) behave very similar, with the CMA-ES usually winning.

Discussion. First of all, the ETP measure seems to work well. It may not be ideal, but at least fulfills the criteria stated as tasks. The tuning potential itself is quite different for the tested algorithm/problem combinations. Sometimes an algorithm is brought from complete failure to robustly finding the optimum, as for the ALT-BFGS on problem 6 with $1E4$ evaluations. The opposite also happens, namely that no better solution is found by the tuning process. This also happens for the ALT-BFGS several times, so it can be assumed that this algorithm is a bit more 'difficult to adapt' than the others. The 2 evolutionary algorithms seem to have the highest tuning potential, followed by Nelder-Mead, and then ALG-BFGS. Note that for technical reasons, it is also possible that the LHS result is seemingly better than the SPO one. The reason is that the LHS table is built with only four runs per configuration, which is seemingly not enough to estimate its quality in some cases. So the LHS configuration may in fact be much better or worse than it first appeared.

5 Conclusions

This work documents several insights. The featured tuning potential measure (ETP) seems to work well, but may still be improved. For instance, the time aspect is not included at all yet. Concerning the comparison of adaptable and non-adaptable algorithms we can state that the potential of the adaptable algorithms is used by tuning if the problems are not already solved to optimality by

the non-adaptable algorithms. Thus, it is a question of time that is needed to adapt a better method. However, the comparison on the CEC test set may be not entirely fair, as the classic methods have not been designed for such problems. Nevertheless, they cope quite well and are only overtaken by well adapted algorithms and/or on the most difficult problems.

References

1. Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size. In: McKay, B., et al. (eds.) Proc. 2005 Congress on Evolutionary Computation (CEC 2005), Piscataway NJ, pp. 1769–1776. IEEE Press, Los Alamitos (2005)
2. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation – The New Experimentalism. Natural Computing Series. Springer, Berlin (2006)
3. Bartz-Beielstein, T., Preuss, M.: Considerations of Budget Allocation for Sequential Parameter Optimization (SPO). In: Paquete, L., Chiarandini, M., Basso, D. (eds.) Empirical Methods for the Analysis of Algorithms, Workshop EMAA 2006, Proceedings, Reykjavik, Iceland, pp. 35–40 (2006)
4. Belisle, C.J.P.: Convergence theorems for a class of simulated annealing algorithms on \mathbb{R}^d . Annals of Applied Probability 29, 885–895 (1992)
5. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
6. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Proc. Genetic and Evolutionary Computation Conf. (GECCO 2002). Morgan Kaufmann, San Francisco (2002)
7. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16, 1190–1208 (1995)
8. Clausen, A.: An implementation of the bfgs algorithm for smooth function minimization (2007), <http://www.econ.upenn.edu/~clausen/computing/bfgs.zip>
9. Eiben, A.E., Jelasity, M.: A critical note on experimental research methodology in EC. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 582–587. IEEE Press, Los Alamitos (2002)
10. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. IEEE Computational Intelligence Magazine 9(2), 159–195 (2001)
11. Jong, K.D.: Parameter setting in eas: a 30 year perspective. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithms. Springer, Berlin (2007)
12. Nash, J.C.: Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation, 2nd edn. IOP, Bristol (1990)
13. Nelder, J.A., Mead: A simplex method for function minimization. The Computer Journal (7), 308–313 (1965)
14. Schwefel, H.-P.: Direct search for optimal parameters within simulation models. In: Conine, R.D., et al. (eds.) Proc. Twelfth Annual Simulation Symp., Tampa FL, Long Beach, CA, pp. 91–102. IEEE Computer Society, Los Alamitos (1979)
15. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore (May 2005), <http://www.ntu.edu.sg/home/EPNSugan>

A Stigmergy-Based Algorithm for Continuous Optimization Tested on Real-Life-Like Environment

Peter Korošec and Jurij Šilc

Jožef Stefan Institute, Computer Systems Department
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
{peter.korosec,jurij.silc}@ijs.si

Abstract. This paper presents a solution to the global optimization of continuous functions by the Differential Ant-Stigmergy Algorithm (DASA). The DASA is a newly developed algorithm for continuous optimization problems, utilizing the stigmergic behavior of the artificial ant colonies. It is applied to the high-dimensional real-parameter optimization with low number of function evaluations. The performance of the DASA is evaluated on the set of 25 benchmark functions provided by CEC'2005 Special Session on Real Parameter Optimization. Furthermore, non-parametric statistical comparisons with eleven state-of-the-art algorithms demonstrate the effectiveness and efficiency of the DASA.

Keywords: Bio-inspired algorithm, continuous parameter optimization, post-hoc statistical test, stigmergy.

1 Introduction

In recent years numerous stochastic optimization algorithms have been proposed to solve real-parameter optimization problems, such as controlled random search, simulated annealing, particle swarm optimization, tabu search, differential evolution, real-parameter genetic algorithms, evolution strategies, and most recently ant-colony optimization (ACO).

Although ACO has been proven to be one of the best metaheuristics in some combinatorial optimization problems, the application to the real-parameter optimizations appears more challenging, since the pheromone laying method is not straightforward. There are several possibilities to use ACO for continuous cost function optimization.

- Simplified direct simulation of real ants behavior: Continuous Ant Colony Optimization (CACO) [5,36], CACO-DE [16], API (named after *Pachycondyla apicalis*) [24], Continuous Interacting Ant Colony (CIAC) [9].
- Extend ACO metaheuristic to explore continuous space with discretization: Adaptive Ant Colony Algorithm (AAC) [22], Binary Ant System (BAS) [19], Multilevel Ant-Stigmergy Algorithm (MASA) [20].
- Extend ACO metaheuristic to explore continuous space with probabilistic sampling: Extended Ant Colony Optimization ($ACO_{\mathbb{R}}$) [31], Continuous Ant

- Colony System (CACS) [27], Aggregation Pheromone System (APS) [35], Direct ACO (DACO) [18], Differential Ant-Stigmergy Algorithm (DASA) [21]. – Hybrid methods: ACO_{IR} with Levenberg-Marquard algorithm [32], ACO with Powell search method [13], ACO with Genetic Algorithm (HACO) [7], ACO with tabu-search method [14], Orthogonal Scheme ACO (OSACO) [15], CACO with Immune Algorithm [11].

The remainder of this paper is organized as follows: Section 2 introduces the optimization algorithm called the Differential Ant-Stigmergy Algorithm. Section 3 presents the experimental evaluation on high-dimensional benchmark functions and gives the comparison of the algorithm to some recent evolutionary algorithms. Finally, Section 4 conclude the paper.

2 The Differential Ant-Stigmergy Algorithm (DASA)

2.1 Problem Definition

Real-parameter optimization is an important issue in many areas of human activity. The general problem is to find a vector of parameter values, $\mathbf{x} = (x_1, x_2, \dots, x_D)$, that minimizes a function, $f(\mathbf{x})$, of D real variables, i.e.,

$$\text{Find: } \mathbf{x}^* \mid f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^D.$$

To solve this problem, we created a fine-grained discrete form of continuous domain. With it we were able to represent this problem as a graph. The parameters' differences assigned to the graph vertices are used as a means to move through the search space. This enabled us to use ant-based approach for solving numerical optimization problems.

2.2 Problem Representation

Let x'_i be the current value of the i -th parameter. During the search for the optimal parameter value, the new value, x_i , is assigned to the i -th parameter as follows:

$$x_i = x'_i + \delta_i. \quad (1)$$

Here, δ_i is the so-called *parameter difference* and is chosen from the set $\Delta_i = \Delta_i^- \cup \{0\} \cup \Delta_i^+$, where $\Delta_i^- = \{\delta_{i,k}^- \mid \delta_{i,k}^- = -b^{k+L_i-1}, k = 1, 2, \dots, d_i\}$ and $\Delta_i^+ = \{\delta_{i,k}^+ \mid \delta_{i,k}^+ = b^{k+L_i-1}, k = 1, 2, \dots, d_i\}$. Here $d_i = U_i - L_i + 1$. Therefore, for each parameter x_i , the parameter difference, δ_i , ranges from b^{L_i} to b^{U_i} , where b is the *discrete base*, $L_i = \lfloor \lg_b(\epsilon_i) \rfloor$, and $U_i = \lfloor \lg_b(\max(x_i) - \min(x_i)) \rfloor$. With the parameter ϵ_i , the maximum precision of the parameter x_i is set. This precision is limited by the computer's floating-point arithmetics. To enable a more flexible movement over the search space, the weight ω is added to Eq. 1:

$$x_i = x'_i + \omega \delta_i \quad (2)$$

where ω is random integer ranges from 1 to $b - 1$.

From all the sets Δ_i , $1 \leq i \leq D$, where D represents the number of parameters, a *search graph* $\mathcal{G} = (V, E)$ with a set of vertices, V , and a set of edges, E , between the vertices is constructed. Each vector Δ_i is represented by the set of vertices, $V_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,2d_i+1}\}$, and $V = \bigcup_{i=1}^D V_i$. Then we have that $\Delta_i = \{\delta_{i,d_i}^-, \dots, \delta_{i,d_i-j+1}^-, \dots, \delta_{i,1}^-, 0, \delta_{i,1}^+, \dots, \delta_{i,j}^+, \dots, \delta_{i,d_i}^+\}$ corresponds to $V_i = \{v_{i,1}, \dots, v_{i,j}, \dots, v_{i,d_i+1}, \dots, v_{i,d_i+1+j}, \dots, v_{i,2d_i+1}\}$, where $v_{i,j} \xrightarrow{\delta} \delta_{i,d_i-j+1}^-$, $v_{i,d_i+1} \xrightarrow{\delta} 0$, $v_{i,d_i+1+j} \xrightarrow{\delta} \delta_{i,j}^+$, and $j = 1, 2, \dots, d_i$. Each vertex of the set V_i is connected to all the vertices that belong to the set V_{i+1} . Therefore, this is a directed graph, where each path \mathbf{p} from the starting vertex, $v_1 \in V_1$, to any of the ending vertices, $v_D \in V_D$, is of equal length and can be defined with v_i as $\nu = (v_1 v_2 \dots v_i \dots v_D)$, where $v_i \in V_i$, $1 \leq i \leq D$.

2.3 Algorithm Implementation

The optimization consists of an iterative improvement of the temporary best solution, \mathbf{x}^{tb} , by constructing an appropriate path \mathbf{p} . New solutions are produced by applying \mathbf{p} to \mathbf{x}^{tb} (Eq. 2).

First a solution \mathbf{x}^{tb} is randomly chosen and evaluated. Then a search graph is created (see Section 2.2) and an initial amount of pheromone is deposited on search graph according to the Cauchy probability density function

$$C(z) = \frac{1}{s\pi(1 + (\frac{z-l}{s})^2)},$$

where l is the location offset and $s = s_{\text{global}} - s_{\text{local}}$ is the scale factor. For an initial pheromone distribution the standard Cauchy distribution ($l = 0$, $s_{\text{global}} = 1$, and $s_{\text{local}} = 0$) is used and each parameter vertices are equidistantly arranged between $z = [-4, 4]$.

There are m ants in a colony, all of which begin simultaneously from the starting vertex. Ants use a probability rule to determine which vertex will be chosen next. The rule is based on the Simple-ACO [33]. More specifically, ant a in step i moves from a vertex in set V_{i-1} to vertex $v_{i,j} \in V_i$ with a probability given by:

$$\text{prob}(a, v_{i,j}) = \frac{\tau(v_{i,j})}{\sum_{1 \leq k \leq 2d_i+1} \tau(v_{i,k})},$$

where $\tau(v_{i,k})$ is the amount of pheromone in vertex $v_{i,k}$.

The ants repeat this action until they reach the ending vertex. For each ant i , path \mathbf{p}_i is constructed. If for some predetermined number of tries (in our case m^2) we get $\mathbf{p}_i = \mathbf{0}$ the search process is reset by randomly choosing new \mathbf{x}^{tb} and pheromone re-initialization. Otherwise, a new solution \mathbf{x}_i is constructed.

After all ants have created solutions, they are being evaluated with a calculation of $f(\mathbf{x}_i)$. If a new solution is better than current best the information about the new solution is stored.

The current best solution, \mathbf{x}^{cb} is compared to the temporary best solution \mathbf{x}^{tb} . If \mathbf{x}^{cb} is better than \mathbf{x}^{tb} , then \mathbf{x}^{tb} values are replaced with \mathbf{x}^{cb} values.

In this case s_{global} is increased according to the global scale increase factor, s_+ : $s_{\text{global}} \leftarrow (1 + s_+)s_{\text{global}}$, s_{local} is set to $s_{\text{local}} = \frac{1}{2}s_{\text{global}}$ and pheromone amount is redistributed according to the associated path \mathbf{p}^{cb} . Furthermore, if new \mathbf{x}^{tb} is better than \mathbf{x}^{b} , then \mathbf{x}^{b} values are replaced with \mathbf{x}^{tb} values. So, global best solution is stored. If no better solution is found s_{global} is decreased according to the global scale decrease factor, s_- : $s_{\text{global}} \leftarrow (1 - s_-)s_{\text{global}}$.

Pheromone evaporation is defined by some predetermined percentage ρ . The probability density function $C(z)$ is changed in the following way: $l \leftarrow (1 - \rho)l$ and $s_{\text{local}} \leftarrow (1 - \rho)s_{\text{local}}$. Here we must emphasize that $\rho > s_-$, because otherwise we might get negative scale factors.

The whole procedure is then repeated until some ending condition is met.

3 Performance Evaluation

3.1 The Experimental Environment

The computer platform used to perform the experiments was based on AMD OpteronTM 2.6-GHz processor, 2 GB of RAM, and the Microsoft[®] Windows[®] XP 32-bit operating system. The DASA was implemented in Borland[®] DelphiTM and for the benchmark suite the original implementation of functions were used in a form of dynamic link library.

3.2 The Benchmark Suite

The DASA algorithm was tested on 25 benchmark functions provided for the CEC'2005 Special Session on Real Parameter Optimization [34]:

- 5 unimodal functions (f_1 : Sphere function displaced, f_2 : Schwefel's problem 1.2 displaced, f_3 : Elliptical function rotated widely conditioned, f_4 : Schwefel's problem 1.2 displaced with noise in the fitness, f_5 : Schwefel's problem 2.6 with global optimum in the frontier).
- 20 multimodal functions
 - 7 basic functions (f_6 : Rosenbrock function displaced, f_7 : Griewank function displaced and rotated without frontiers, f_8 : Ackley function displaced and rotated with the global optimum in the frontier, f_9 : Rastrigin function displaced, f_{10} : Rastrigin function displaced and rotated, f_{11} : Weierstrass function displaced and rotated, and f_{12} : Schwefel's problem 2.13).
 - 2 expanded functions (f_{13} and f_{14}).
 - 11 hybrid functions (f_{15} to f_{25}). Each one of them have been defined through compositions of 10 out of the 14 previous functions (different in each case).

All functions have been displaced in order to ensure that their optima can never be found in the center of the search space. In two functions, in addition, the optima can not be found within the initialization range, and the domain of search is not limited (the optimum is out of the range of initialization).

Here, 30-dimensional functions were optimized using only 1000 evaluations¹ (to simulate a real-life-like environment) with 25 runs on each function. However, the DASA was already tested on some CEC'2005 benchmark functions with 300.000 evaluations. Comparison with some selected algorithms showed that the DASA provides highly competitive results [21].

3.3 Parameter Settings

The DASA has three parameters: the number of ants, m , the pheromone evaporation factor, ρ , and the maximum parameter precision, ϵ . Their settings are: $m = 10$, $\rho = 0.1$, and $\epsilon = 10^{-15}$. We must note that during the experimentation we did not fine-tune the algorithms parameters, but only carried out a limited number of experiments to find satisfying settings.

3.4 Compared Algorithms

The DASA is compared to the eleven algorithms which were presented in the CEC'2005 Special Session on Real Parameter Optimization. For more details on the description and parameters used for each one, please refer to the respective contributions. The algorithms are: a hybrid real-coded genetic algorithm with female and male differentiation (BLX-GL50) [12], a real-coded memetic algorithm with adaptive local search parameters (BLX-MA) [23], an evolutionary algorithm with mutation step co-evolution (CoEVO) [26], a differential evolution (DE) [28], a continuous estimation of distribution algorithm (EDA) [37], a flexible evolutionary algorithm (FEA) [1], a restart covariance matrix adaptation evolutionary strategy with increasing population size (G-CMA-ES) [2], a guided differential evolution (G-DE) [6], a population-based steady-state procedure (K-PCX) [30], an advanced local search evolutionary algorithm (L-CMA-ES) [3], and a steady-state real-parameter genetic algorithm (SPC-PNX) [4].

3.5 Comparison

For each function and each algorithm the error value (Error) is collected for 25 runs and the trials mean are presented in Table 1.

In what follows, we analyze the performance of the DASA algorithm with respect to the remaining ones, and determine if the results it offers are better than the ones offered by the rest of algorithms. For this multiple-function analysis we use a non-parametric statistical test which is less restrictive than parametric ones (e.g., repeated-measures ANOVA) and they can be used over small size samples of results [8].

Performance comparison can be initiated after the ranking trial run results according to their relative performance. Under the null hypothesis, the k algorithms are equivalent. If the null hypothesis is rejected then at least one of the k algorithms performed differently from at least one other algorithm. However, this does not indicate which one. To get this information a *post-hoc* test is used.

¹ At the CEC'2005 Special Session on Real Parameter Optimization the maximum number of function evaluations was set to 300.000!

Table 1. Mean error values achieved with 12 algorithms for 25 functions with dimensions 30 after 1000 FEs

	BLX-GL50	BLX-MA	CoEVO	DE	EDA	FEA
f_1	4.80E+4	2.08E+4	6.68E+4	2.06E+4	8.58E+4	2.69E+4
f_2	8.58E+4	5.53E+4	9.26E+4	7.72E+4	1.12E+5	6.25E+4
f_3	7.47E+8	2.94E+8	1.19E+9	5.53E+8	1.25E+9	5.05E+8
f_4	1.07E+5	1.15E+5	1.23E+5	9.40E+4	1.29E+5	7.41E+4
f_5	2.78E+4	2.86E+4	3.78E+4	2.62E+4	3.79E+4	2.48E+4
f_6	2.00E+0	5.57E+9	3.81E+0	14.3E+9	59.7E+9	9.42E+9
f_7	6.07E+3	3.16E+3	4.80E+3	6.94E+3	1.09E+4	2.73E+3
f_8	2.12E+1	2.09E+1	2.12E+1	2.12E+1	2.12E+1	2.12E+1
f_9	3.84E+2	2.99E+2	4.31E+2	3.77E+2	4.80E+2	3.06E+2
f_{10}	5.51E+2	5.17E+2	6.84E+2	5.01E+2	7.88E+2	4.90E+2
f_{11}	4.57E+1	3.74E+1	4.48E+1	4.56E+1	4.54E+1	4.61E+1
f_{12}	1.32E+6	4.16E+5	1.61E+6	1.53E+6	1.83E+6	9.55E+5
f_{13}	2.29E+5	3.95E+3	2.58E+2	1.62E+5	7.50E+5	1.53E+4
f_{14}	1.42E+1	1.39E+1	1.41E+1	1.41E+1	1.42E+1	1.42E+1
f_{15}	1.00E+3	7.62E+2	1.12E+3	1.08E+3	1.13E+3	8.83E+2
f_{16}	7.06E+2	6.43E+2	9.75E+2	8.33E+2	9.16E+2	6.21E+2
f_{17}	7.98E+2	1.02E+3	1.10E+3	9.05E+2	9.98E+2	7.35E+2
f_{18}	1.20E+3	1.09E+3	1.33E+3	1.28E+3	1.30E+3	1.18E+3
f_{19}	1.20E+3	1.08E+3	1.31E+3	1.28E+3	1.32E+3	1.17E+3
f_{20}	1.20E+3	1.10E+3	1.33E+3	1.29E+3	1.30E+3	1.20E+3
f_{21}	1.37E+3	1.32E+3	1.43E+3	1.44E+3	1.48E+3	1.30E+3
f_{22}	1.43E+3	1.33E+3	1.63E+3	1.57E+3	1.63E+3	1.38E+3
f_{23}	1.35E+3	1.37E+3	1.43E+3	1.42E+3	1.50E+3	1.32E+3
f_{24}	1.40E+3	1.45E+3	1.49E+3	1.48E+3	1.51E+3	1.36E+3
f_{25}	1.61E+3	1.46E+3	1.57E+3	2.07E+3	1.91E+3	1.40E+3

	G-CMA-ES	G-DE	K-PCX	L-CMA-ES	SPC-PNX	DASA
f_1	8.16E+2	6.42E+4	1.11E+1	4.54E+3	1.99E+4	2.75E+3
f_2	2.39E+5	1.03E+5	9.91E+3	6.38E+4	6.28E+4	6.63E+4
f_3	1.07E+9	8.57E+8	5.13E+7	2.36E+8	4.36E+8	1.57E+8
f_4	1.55E+6	1.14E+5	1.69E+5	4.20E+8	7.35E+4	1.54E+5
f_5	1.07E+4	2.77E+4	4.14E+4	1.89E+4	3.50E+4	2.21E+4
f_6	1.77E+7	30.7E+9	2.39E+6	4.88E+8	36.7E+9	1.24E+8
f_7	1.39E+2	6.00E+3	7.73E+1	8.24E+2	7.75E+3	7.64E+2
f_8	2.12E+1	2.12E+1	2.12E+1	2.12E+1	2.12E+1	2.09E+1
f_9	2.53E+2	4.13E+2	5.03E+2	4.74E+2	4.95E+2	7.47E+1
f_{10}	2.77E+2	6.52E+2	8.36E+2	1.14E+3	6.46E+2	5.85E+2
f_{11}	4.54E+1	4.55E+1	3.30E+1	4.01E+1	4.52E+1	4.09E+1
f_{12}	1.67E+6	1.30E+6	1.05E+5	4.54E+5	1.46E+6	2.10E+5
f_{13}	1.14E+2	5.46E+5	1.29E+6	2.06E+2	8.77E+5	2.32E+3
f_{14}	1.42E+1	1.42E+1	1.44E+1	1.47E+1	1.42E+1	1.41E+1
f_{15}	6.69E+2	9.87E+2	1.02E+3	8.60E+2	9.99E+2	5.48E+2
f_{16}	3.75E+2	7.74E+2	1.20E+3	5.96E+2	7.87E+2	6.30E+2
f_{17}	4.79E+2	8.62E+2	1.36E+3	1.94E+3	8.61E+2	1.06E+3
f_{18}	9.45E+2	1.17E+3	1.42E+3	1.22E+3	1.29E+3	1.10E+3
f_{19}	9.51E+2	1.17E+3	1.42E+3	1.27E+3	1.30E+3	1.11E+3
f_{20}	9.50E+2	1.17E+3	1.42E+3	1.17E+3	1.28E+3	1.04E+3
f_{21}	9.44E+2	1.35E+3	1.56E+3	1.17E+3	1.15E+3	1.20E+3
f_{22}	1.08E+3	1.40E+3	1.91E+3	1.41E+3	1.59E+3	1.47E+3
f_{23}	1.03E+3	1.35E+3	1.56E+3	1.40E+3	1.44E+3	1.19E+3
f_{24}	9.97E+2	1.38E+3	1.59E+3	1.70E+3	1.46E+3	1.45E+3
f_{25}	3.05E+2	1.67E+3	1.69E+3	2.12E+3	1.79E+3	1.41E+3

Table 2. Algorithms ranking

	BLX-GL50	BLX-MA	CoEVO	DE	EDA	FEA	G-CMA-ES	G-DE	K-PCX	L-CMA-ES	SPC-PNX	DASA
f_1	9.0	7.0	11.0	6.0	12.0	8.0	2.0	10.0	1.0	4.0	5.0	3.0
f_2	8.0	2.0	9.0	7.0	11.0	3.0	12.0	10.0	1.0	5.0	4.0	6.0
f_3	8.0	4.0	11.0	7.0	12.0	6.0	10.0	9.0	1.0	3.0	5.0	2.0
f_4	4.0	6.0	7.0	3.0	8.0	2.0	11.0	5.0	10.0	12.0	1.0	9.0
f_5	7.0	8.0	10.0	5.0	11.0	4.0	1.0	6.0	12.0	2.0	9.0	3.0
f_6	8.0	5.0	11.0	7.0	12.0	6.0	2.0	9.0	1.0	4.0	10.0	3.0
f_7	9.0	6.0	7.0	10.0	12.0	5.0	2.0	8.0	1.0	4.0	11.0	3.0
f_8	7.5	1.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	1.5
f_9	6.0	3.0	8.0	5.0	10.0	4.0	2.0	7.0	12.0	9.0	11.0	1.0
f_{10}	5.0	4.0	9.0	3.0	10.0	2.0	1.0	8.0	11.0	12.0	7.0	6.0
f_{11}	11.0	2.0	5.0	10.0	7.5	12.0	7.5	9.0	1.0	3.0	6.0	4.0
f_{12}	7.0	3.0	10.0	9.0	12.0	5.0	11.0	6.0	1.0	4.0	8.0	2.0
f_{13}	8.0	5.0	3.0	7.0	10.0	6.0	1.0	9.0	12.0	2.0	11.0	4.0
f_{14}	7.5	1.0	3.0	3.0	7.5	7.5	7.5	7.5	11.0	12.0	7.5	3.0
f_{15}	8.0	3.0	11.0	10.0	12.0	5.0	2.0	6.0	9.0	4.0	7.0	1.0
f_{16}	6.0	5.0	11.0	9.0	10.0	3.0	1.0	7.0	12.0	2.0	8.0	4.0
f_{17}	3.0	8.0	10.0	6.0	7.0	2.0	1.0	5.0	11.0	12.0	4.0	9.0
f_{18}	6.0	2.0	11.0	8.0	10.0	5.0	1.0	4.0	12.0	7.0	9.0	3.0
f_{19}	6.0	2.0	10.0	8.0	11.0	4.5	1.0	4.5	12.0	7.0	9.0	3.0
f_{20}	6.5	3.0	11.0	9.0	10.0	6.5	1.0	4.5	12.0	4.5	8.0	2.0
f_{21}	8.0	6.0	9.0	10.0	11.0	5.0	1.0	7.0	12.0	3.0	2.0	4.0
f_{22}	6.0	2.0	10.5	8.0	10.5	3.0	1.0	4.0	12.0	5.0	9.0	7.0
f_{23}	4.5	6.0	9.0	8.0	11.0	3.0	1.0	4.5	12.0	7.0	10.0	2.0
f_{24}	4.0	5.5	9.0	8.0	10.0	2.0	1.0	3.0	11.0	12.0	7.0	5.5
f_{25}	6.0	4.0	5.0	11.0	10.0	2.0	1.0	7.0	8.0	12.0	9.0	3.0
Avg	6.76	4.16	8.72	7.38	10.2	4.76	3.58	6.7	8.22	6.36	7.4	3.76
Std	1.82	2.05	2.48	2.27	1.58	2.39	3.91	2.02	4.76	3.69	2.69	2.20

In Table 2 algorithms rankings with average ranks and standard deviations are presented. The DASA ranks second, but it has much lower standard deviation than the first algorithm G-CMA-ES.

For statistical hypothesis test we use Iman-Davenport test [17] based on the following statistic:

$$F_F = \frac{(N - 1)\chi_F^2}{N(k - 1) - \chi_F^2},$$

where k is number of compared algorithms, N is number of benchmark functions, and χ_F^2 is Friedman's statistics [10]. For chosen significance level α (usually 0.05 or 0.10), the null hypothesis is rejected if $\alpha < p\text{-value}$, where $p\text{-value}$ is determined according to F_F statistic and can be found in [29,38].

In our case we have $F_F = 12.0601$ and $p\text{-value}$ is 0.0. Given that the $p\text{-value}$ of Iman-Davenport is lower than the levels of significance considered ($\alpha = 0.05$) the null hypothesis is rejected and there are significant differences among the compared algorithms.

Attending to these results, a post-hoc statistical analysis could help up to detect concrete differences among algorithms. For post-hoc test we use Nemenyi test [25] where the performance of two algorithms is significantly different if the corresponding average of rankings is at least as great as its critical distance, CD :

$$CD = Q_\alpha \sqrt{\frac{k(k+1)}{6N}},$$

where Q_α is the critical value for a multiple non-parametric comparison with a control [38]. Comparison of the algorithms with Nemenyi test is shown in Fig. 1.

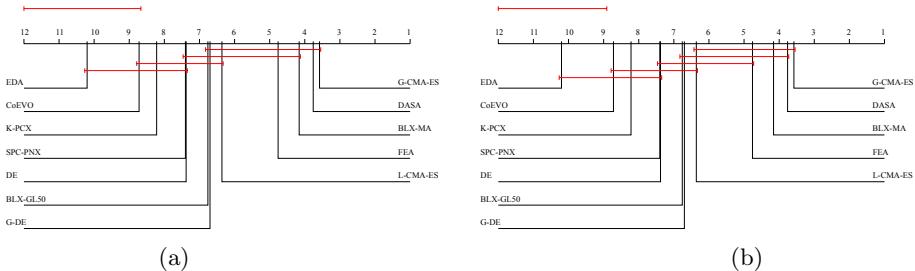


Fig. 1. The Nemenyi *post-hoc* statistical test: (a) $\alpha = 0.05$, $CD = 3.3327$; (b) $\alpha = 0.10$, $CD = 3.0900$

Nemenyi post-hoc test allows us to point out the following differences, considering the DASA as control algorithm: the DASA is better than EDA, CoEvo and K-PCX with $\alpha = 0.05$ (3/12 algorithms) and is also better than SPC-PNX and DE with $\alpha = 0.10$ (5/12 algorithms).

4 Conclusion

In this paper the performance evaluation of the Differential Ant-Stigmergy Algorithm (DASA) was performed on the set of 25 benchmark functions provided by CEC'2005 Special Session on Real Parameter Optimization. Additionally, the DASA was compared by a nonparametric statistical test to the published results for the 11 algorithms submitted to the CEC'2005 Special Session on Real Parameter Optimization. To make this comparison possible only the algorithms that had all the appropriate results for 30-dimensional functions were selected. Most of them are the up-to-date variations of current state-of-the-art algorithms in continuous optimization.

The experimental results have shown that the DASA ranks among the best algorithms for real-life-like applications. Its main advantage is in consistent problem solving which is indicated by 19 rankings in top third, 4 ranking in middle third and only 2 in bottom third. If we compare ranking of f_9 and f_{10} , we can see that the DASA perform much worse on f_{10} , which is a rotated f_9 function. This might indicate the DASA's dependence on coordinate system. On the other hand, the order of problem parameters has no influence on solution quality.

Statistical analysis following the Nemenyi post-hoc test with $\alpha = 0.10$ showed that the DASA is significantly better than 5 out of 11 compared algorithms.

References

1. Alonso, S., Jimenez, J., Carmona, H., Galvan, B., Winter, G.: Performance of a Flexible Evolutionary Algorithm, <http://www.ntu.edu.sg/home/EPNSugan>
2. Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size. In: Proc. CEC 2005, Edinburg, UK, pp. 1769–1776 (2005)
3. Auger, A., Hansen, N.: Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In: Proc. CEC 2005, Edinburg, UK, pp. 1777–1784 (2005)
4. Ballester, P.J., Stephenson, J., Carter, J.N., Gallagher, K.: Real-parameter optimization performance study on the CEC 2005 benchmark with SPC- PNX. In: Proc. CEC 2005, Edinburg, UK, pp. 498–505 (2005)
5. Bilchev, G., Parmee, I.C.: The ant colony metaphor for searching continuous design spaces. LNCS, vol. 993, pp. 25–39. Springer, Heidelberg (1995)
6. Bui, L.T., Shan, Y., Qi, F., Abbass, H.A.: Comparing Two Versions of Differential Evolution in Real Parameter Optimization, <http://www.ntu.edu.sg/home/EPNSugan>
7. Chen, C., Tian, X.Y., Zou, X.Y., Cai, P.X., Mo, J.Y.: A hybrid ant colony optimization for the prediction of protein secondary structure. Chinese Chem. Lett. 16, 1551–1554 (2005)
8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
9. Dréo, J., Siarry, P.: A new ant colony algorithm using the heterarchical concept aimed at optimization of multimimima continuous functions. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 216–221. Springer, Heidelberg (2002)
10. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 32, 675–701 (1937)
11. Gao, W.: Immunized continuous ant colony algorithm. In: Proc. 26th Chinese Control Conf., Zhangjiajie, China, pp. 705–709 (2007)
12. García-Martínez, C., Lozano, M.: Hybrid Real-Coded genetic Algorithm with Female and Male Differentiation. In: Proc. CEC 2005, Edinburg, UK, pp. 896–903 (2005)
13. Ge, Y., Meng, Q.C., Yan, C.J., Xu, J.: A hybrid ant colony algorithm for global optimization of continuous multi-extreme functions. In: Proc. 3rd Int. Conf. Machine Lear. Cyber., Shanghai, China, pp. 2427–2432 (2004)
14. Ho, S.L., Yang, S., Ni, G., Machado, J.M.: A modified ant colony optimization algorithm modeled on tabu-search methods. IEEE T. Magn. 42, 1195–1198 (2006)
15. Hu, X.M., Zhang, J., Li, Y.: Orthogonal methods based ant colony search for solving continuous optimization problems. J. Comput. Sci. Technol. 23, 2–18 (2008)
16. Huang, H., Hao, Z.: ACO for continuous optimization based on discrete encoding. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 504–505. Springer, Heidelberg (2006)
17. Iman, R.L., Davenport, J.M.: Approxymations of the critical region of the Friedman statistic. Commun. Stat. 9, 571–595 (1980)
18. Kong, M., Tian, P.: A direct application of ant colony optimization to function optimization problem in continuous domain. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 324–331. Springer, Heidelberg (2006)
19. Kong, M., Tian, P.: A binary ant colony optimization for the unconstrained function optimization problem. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS, vol. 3801, pp. 682–687. Springer, Heidelberg (2005)

20. Korošec, P., Šilc, J.: The multilevel ant stigmergy algorithm: An industrial case study. In: Proc. 7th Int. Conf. Comput. Intell. Natural Comput, Salt Lake City, UT, pp. 475–478 (2005)
21. Korošec, P., Šilc, J., Oblak, K., Kosel, F.: The differential ant-stigmergy algorithm: An experimental evaluation and a real-world application. In: Proc. CEC 2007, Singapore, pp. 157–164 (2007)
22. Li, Y.J., Wu, T.J.: An adaptive ant colony system algorithm for continuous-space optimization problems. *J. Zhejiang Univ. - Sc. A* 4, 40–46 (2003)
23. Molina, D., Herrera, F., Lozano, M.: Adaptive Local Search Parameters for Real-Coded Memetic Algorithms. In: Proc. CEC 2005, Edinburg, UK, pp. 888–895 (2005)
24. Monmarché, V.G., Slimane, M.: On how pachycondyla apicalis ants suggest a new search algorithm. *Future Gener. Comp. Sy.* 16, 937–946 (2000)
25. Nemenyi, P.B.: Distribution-free Multiple Comparison. PhD Thesis, Princeton University (1963)
26. Pošík, P.: Real-Parameter Optimization Using the Mutation Step Co-Evolution. In: Proc. CEC 2005, Edinburg, UK, pp. 872–879 (2005)
27. Pourtakdoust, S.H., Nobahari, H.: An extension of ant colony system to continuous optimization problems. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 294–301. Springer, Heidelberg (2004)
28. Rönkkönen, J., Kukkonen, S., Price, K.V.: Real-Parameter Optimization Using the Mutation Step Co-Evolution. In: Proc. CEC 2005, Edinburg, UK, pp. 506–513 (2005)
29. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2000)
30. Sinha, A., Tiwari, S., Deb, K.: A Population-Based, Steady-State Procedure for Real-Parameter Optimization. In: Proc. CEC 2005, Edinburg, UK, pp. 514–521 (2005)
31. Socha, K.: ACO for continuous and mixed-variable optimization. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 25–36. Springer, Heidelberg (2004)
32. Socha, K., Blum, C.: An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. *Neural Comput. Appl.* 16, 235–247 (2007)
33. Stützle, T., Dorigo, M.: An experimental study of the simple ant colony optimization algorithm. In: Proc. WSES Int. Conf. Evol. Comput., Tenerife, Spain, pp. 253–258 (2001)
34. Suganthan, P.N., Hansen, N., Liang, J.J., Chen, Y.P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical Report, Nanyang Technological University, Singapore (May 2005), <http://www.ntu.edu.sg/home/EPNSugan>
35. Tsutsui, S.: Aggregation pheromone system: A real-parameter optimization algorithm using aggregation pheromones as the base metaphor. *T. Jpn. Soc. Artif. Intell.* 20, 76–83 (2005)
36. Wodrich, M., Bilchev, G.: Cooperative distributed search: The ant's way. *Control Cybern.* 26, 413–446 (1997)
37. Yuan, B., Gallagher, M.: Experimental Results for the Special Session on Real-Parameter Optimization at CEC 2005: A Simple, Continuous EDA. In: Proc. CEC 2005, Edinburg, UK, pp. 1792–1799 (2005)
38. Zar, J.H.: Biostatistical Analysis. Prentice-Hall, Englewood Cliffs (1999)

Stochastic Local Search Techniques with Unimodal Continuous Distributions: A Survey

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic
posik@labe.felk.cvut.cz

Abstract. In continuous black-box optimization, various stochastic local search techniques are often employed, with various remedies for fighting the premature convergence. This paper surveys recent developments in the field (the most important from the author's perspective), analyzes the differences and similarities and proposes a taxonomy of these methods. Based on this taxonomy, a variety of novel, previously unexplored, and potentially promising techniques may be envisioned.

1 Introduction

Stochastic local search (SLS) methods originated in the field of combinatorial optimization and they are claimed to be among the most efficient optimizers. In [1] they are informally described as ‘local search algorithms that make use of randomized choices in generating or selecting candidate solutions for a given combinatorial problem instance.’ As noted in [2], ‘once randomized and appended or hybridized by a local search component, these (SLS techniques) include a wealth of methods such as simulated annealing, iterated local search, greedy randomized adaptive search, variable neighbourhood search, ant colony optimization, among others,’ which are usually classified as global search heuristics. The *locality* is usually induced by the perturbation operators used to generate new solutions.

In this paper, the term *stochastic local search* is used to describe algorithms for continuous optimization as well. The local neighborhood is often not given by a perturbation operator, but rather by a single-peak probability distribution function (p.d.f) with decaying tails (very often Gaussian). Even though generally the value of p.d.f.s non-zero for any point in the feasible space, the offspring are concentrated ‘near’ the distribution center. Due to this fact, many of these algorithms exhibit a kind of hill-climber behaviour, which is, according to the author, sufficient to describe them as SLS.

The algorithms discussed in this article arised mainly from two sources: evolutionary strategies and estimation of distribution algorithms. Evolution strategies (ES) (see e.g. [3] for recent introduction) were among the first algorithms for continuous black-box optimization which employed a stochastic component. Their first versions were purely mutative and used $(1+1)$, or $(1+\lambda)$ selection operators. The individual solutions were coupled with the distribution parameters which underwent the evolution along with the

candidate solutions. Later, they were generalized to $(\mu + \lambda)$ -ES which were still only mutative, but allowed the search to take place in several places of the search space in parallel. With μ parents, it became possible to introduce the crossover operator which is considered to be the main evolutionary search operator in the field of genetic algorithms (GA) [4], and the multi-recombinant ES were born [5]. The notation $(\mu/\rho + \lambda)$ -ES means that out of the μ parents, ρ of them were recombined to become a center for one of the λ generated offspring. After another two decades of research, the state-of-the-art evolutionary strategy with covariance matrix adaptation (CMA-ES) was developed [6]. Rather surprisingly, it is a kind of $(\mu/\mu, \lambda)$ -ES—the computation of the center of the normal distribution is based on all selected parents, i.e. the same distribution is used to generate all candidate solutions. Even though the CMA-ES is a multi-recombinant ES, in each generation it searches the neighborhood of 1 point and thus exhibits local search behaviour (given the step size is small compared to the size of the search space which is often the case).

The second source of SLS lies in estimation-of-distribution algorithms (EDAs) [7]. There are many variants that use multimodal distributions, but here we are concerned with unimodal ones. The first continuous EDAs modeled all variables independently (e.g. [8], [9]). EDAs using full covariance matrix [10] and EDAs using Bayesian factorization of the normal distribution [11] emerged shortly thereafter. These EDAs used almost exclusively maximum-likelihood estimates (MLE) of the distribution parameters—an approach that was very successful in case of discrete EDAs. However, it turned out very soon ([12], [13], [14], [15], [16]) that MLE leads in case of normal distribution to premature convergence even if the population is situated on the slope of the fitness function! Various remedies of this problem emerged ([17], [18], [19] [20], [21]).

In both areas, ES and EDAs, articles discussing the use of solutions discarded by the selection can be found. The discarded solutions can be used to speed up the adaptation of covariance matrix ([22], [23]), or a completely novel method of learning the distribution can be constructed [20].

As already stated, all the above mentioned algorithms can be described as instances of stochastic local search. In the next section, these key works in this field are surveyed in greater detail. In section 3 the taxonomy of these methods is constructed based on their commonalities and differences. Section 4 proposes a few new possibilities offered by the taxonomy and section 5 concludes the paper.

2 Overview of Selected SLS Techniques

A detailed, thorough, and in-depth comparison of some algorithms relevant to this paper can be found in [24]. This article, on the other hand, is aimed especially at describing the main distinguishing features of more recent algorithms.

A general continuous SLS algorithm with single-peak distribution can be described in high-level as Alg. 1. This formulation can accommodate

- *comma* (generational) and *plus* (steady-state) evolutionary schemes,
- use of *selected and/or discarded* solutions in the model adaptation phase,
- model *adaptation* (the previous model, $\mathcal{M}^{(g-1)}$, enters the `Update` phase),
- *self-adaptation* (the model parameters can be part of the population $X^{(g-1)}$),

Algorithm 1. Continuous Stochastic Local Search

```

1 begin
2    $\mathcal{M}^{(0)} \leftarrow \text{InitializeModel}()$ 
3    $X^{(0)} \leftarrow \text{Sample}(\mathcal{M}^{(0)})$ 
4    $f^{(0)} \leftarrow \text{Evaluate}(X^{(0)})$ 
5    $g \leftarrow 1$ 
6   while not TerminationCondition() do
7      $\{\mathcal{S}, \mathcal{D}\} \leftarrow \text{Select}(X^{(g-1)}, f^{(g-1)})$ 
8      $\mathcal{M}^{(g)} \leftarrow \text{Update}(g, \mathcal{M}^{(g-1)}, X^{(g-1)}, f^{(g-1)}, \mathcal{S}, \mathcal{D})$ 
9      $X_{\text{Offs}} \leftarrow \text{Sample}(\mathcal{M}^{(g)})$ 
10     $f_{\text{Offs}} \leftarrow \text{Evaluate}(X_{\text{Offs}})$ 
11     $\{X^{(g)}, f^{(g)}\} \leftarrow \text{Replace}(X^{(g-1)}, X_{\text{Offs}}, f^{(g-1)}, f_{\text{Offs}})$ 
12     $g \leftarrow g + 1$ 
13 end

```

- *deterministic* model adaptation (a predefined schedule depending on the generation counter g can be used for the model parameters), and
- *feedback* model adaptation (the information on the current population state can be used to adapt the model).

Individual algorithms mentioned in the introduction can all be described in this framework. They will generally differ in the definition of the model, \mathcal{M} , and in the operations `Update` and `Sample`.

Stochastic hill-climbing with learning by vectors of normal distributions (SHCL-VND) [8] uses normal distribution for sampling. The model has the form of $\mathcal{M} = \{\mu, \sigma\}$. In the update phase, it uses a Hebbian learning rule to adapt the center of the distribution, $\mu^{(g)} = \mu^{(g-1)} + \mu_{\text{move}}(\bar{X}_{\mathcal{S}} - \mu^{(g-1)})$, so that the new center is between the old center and the mean of the selected individuals, $\bar{X}_{\mathcal{S}}$, or possibly behind the mean. The spread of the distribution is adapted using deterministic schedule as $\sigma^{(g)} = c_{\text{reduce}}\sigma^{(g-1)}$, $c_{\text{reduce}} \in (0, 1)$.

Univariate marginal distribution algorithm for continuous domains (UMDA_C) [9] also does not take into account any dependencies among variables. This algorithm performs some statistical tests in order to determine which of the theoretical density functions fits the particular variable best.

Maximum-likelihood Gaussian EDA (ML-G-EDA) uses a Gaussian distribution with full covariance matrix Σ to generate new candidate solutions. Its model has the form of $\mathcal{M} = \{\mu, \Sigma\}$. Model is not adapted, it is created from scratch as ML estimate based on the selected individuals only. The update step is thus $\mu^{(g)} = \bar{X}_{\mathcal{S}}$ and $\Sigma^{(g)} = \text{covmat}(X_{\mathcal{S}})$.

This algorithm is highly prone to premature convergence ([12], [13], [14], [15], [16]). To improve the algorithm, several approaches were suggested.

Variance scaling (VS) [13] is the most simple approach for ML-G-EDA improvement. The change is in the sampling phase: the covariance matrix Σ is substituted with

enlarged one, $c\boldsymbol{\Sigma}$, $c \geq 1$. In [25] it was shown that this approach with multivariate normal distribution leads in higher-dimensional spaces either to premature convergence on the slopes of the fitness function, or to the divergence of the distribution in the neighborhood of the optimum.

Adaptive variance scaling (AVS) [17] coupled with ML-G-EDA uses the model in the following form: $\mathcal{M} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, c_{AVS}, f_{BSF}\}$. The covariance matrix enlargement factor c_{AVS} becomes part of the model and is adapted on the basis if the best-so-far (BSF) solution was improved. In the update phase, c_{AVS} is increased ($c_{AVS}^{(g)} = \eta \cdot c_{AVS}^{(g-1)}$) if the best solution in X_S is of better quality than f_{BSF} , or decreased ($c_{AVS}^{(g)} = c_{AVS}^{(g-1)} / \eta$), otherwise.

Correlation trigger (CT) was introduced in the same article [17] as AVS. It was observed that the AVS slows down the algorithm convergence in situations when the distribution is centered around the optimum of the fitness function. In that cases, the c_{AVS} multiplier is too high and it takes several generations to decrease it to reasonable values. A better approach is to trigger the AVS only when the population is on the slope, otherwise pure MLE of variance is used. The rank correlation between the values of probability density function (p.d.f) and fitness values was used as the AVS trigger. If the population is on the slope, the correlation will be low, while in the valley the absolute value of correlation will be high (assuming minimization, with decreasing value of p.d.f the fitness increases).

Standard-deviation ratio (SDR) [18] was later used instead of CT which fails in higher-dimensional spaces. SDR triggers AVS in cases when the improvements are found far away from the distribution center (if the distance of the average of all improving solutions in the current generation is larger than a threshold).

Anticipated mean shift (AMS) [21] is another scheme for fighting the premature convergence. In fact, it belongs to this article only partially: the parameters of the distribution are estimated as in the case of single-peak Gaussian, however, in the sampling phase 2-peak distribution is used (with the same shape parameters, but different means). This way, part of the offspring is artificially moved in the direction of estimated gradient (anticipated mean shift). It is assumed that if this prediction was right, then the shifted solutions will be selected along with some of the non-shifted solutions which in turn increases the variance in the direction of the gradient.

Other than normal distributions were explored in several works. In [26], anisotropic Cauchy distribution was used to fasten ES, but the algorithm actually exploits separability of the problem as shown in [27],[28]. In [19], the Gaussian, isotropic Gaussian, and isotropic Cauchy distributions were compared from the point of view if non-adaptive variance scaling (VS) is sufficient to preserve the needed diversity. The isotropic Cauchy distribution was the most promising. The shape and the center of the distribution were estimated using MLE for the Gaussian distribution in all cases. In subsequent experiments it turned out that this approach fails e.g. on ridge functions.

Evolutionary strategy with covariance matrix adaptation (CMA-ES) [6] is currently considered the state-of-the-art technique in numerical optimization. This algorithm nowadays exists in several variants, but all have some common features. Detailed

description of CMA-ES is beyond the scope of this paper; note that its model is given by $\mathcal{M} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, c, \mathbf{p}_c, \mathbf{p}_\sigma\}$. CMA-ES differs from other approaches

1. by using a kind of aggregated memory, the so called evolution paths \mathbf{p}_c and \mathbf{p}_σ , which are cumulated over generations and used to adapt $\boldsymbol{\Sigma}$ and c , and
2. by estimating the distribution of selected mutation steps, rather than distribution of selected individuals.

CMA-ES using also the discarded individuals was proposed in [22], and further discussed in [23]. The covariance matrix adaptation mechanism uses also the discarded individuals with negative weights. The covariance matrix $\boldsymbol{\Sigma}$ might lose its positive definiteness, but in practice it does not happen. Speedups of the order of 2 were observed using this strategy in high-dimensional spaces with large populations.

Optimization via classification was explored in the context of single-Gaussian-based SLS in [20]. Both the selected and the discarded individuals are used to train a classifier that distinguishes between them. If the classifier has a suitable structure (quadratic discriminant function), it can be transformed into a probabilistic model (Gaussian). Similar idea was used before in discrete space [29], or in continuous spaces [30], but the classifier in that case was not transformable to a single peak distribution and is thus out of the scope of this article.

Adaptive encoding (AE) [31] is not directly an optimization algorithm. In continuous domain, the ability to find the right rotation of the search space is crucial. AE decouples the space transformation part from the CMA-ES and makes it available for any search algorithm, especially for the single-peak SLS. To decouple the transformation part from the optimization algorithm was proposed also in other works, e.g. in [32].

3 Continuous SLS Taxonomy

The algorithms that were just described can be categorized from many points of view. These features are (to a great extent) independent of each other and new algorithms can be constructed just by deciding on each feature.

3.1 Model Sampling

One of the most important aspects of any algorithm is the choice of the sampling distribution \mathcal{P} . The sampling process then reads

$$\mathbf{z}_i \sim \mathcal{P}, \quad (1)$$

$$\mathbf{x}_i = \boldsymbol{\mu} + c \cdot \mathbf{R} \times \text{diag}(\boldsymbol{\sigma}) \times \mathbf{z}_i. \quad (2)$$

Here it is assumed that single-peak origin-centered base distributions \mathcal{P} (non-parametric, or with fixed parameters) are used to sample new raw candidate solutions, \mathbf{z}_i . The distribution is enlarged as a whole by multiplying it with a global step size c and elongated along the coordinate axes by multiplying each d th coordinate with the respective multiplicator σ_d ($\text{diag}(\boldsymbol{\sigma})$ is a diagonal matrix with entries σ_d on the diagonal). The

sampled points are rotated by using the rotation matrix \mathbf{R} with orthonormal columns. The origin of the distribution is then changed by adding the position vector $\boldsymbol{\mu}$. The model parameters $\boldsymbol{\mu}$, c , \mathbf{R} , and $\boldsymbol{\sigma}$ are created in the model building phase. The base distribution \mathcal{P} is usually fixed during the whole evolution.

Regarding the model sampling, the individual algorithms can differ in the following aspects:

Feature 1. *What kind of base distribution \mathcal{P} is used for sampling?*

The majority of algorithms use standard Gaussian distribution. In [19], scaled versions of isotropic Gaussian and isotropic Cauchy distributions¹ were analyzed. The distributions were scaled by a constant so that if the distribution was centered around the optimum of a sphere function, then after selecting τN best individuals, the distance of the furthest is expected to be 1.

Feature 2. *Is the type of distribution fixed during the whole evolution?*

Switching the types of probabilistic models is not quite common, but was already employed on the basis of individual axes [9]. It is also possible to change the type of model as a whole.

3.2 Model Building

In the phase of model building, there are two main tasks

1. set the model parameters $\boldsymbol{\mu}$, c , \mathbf{R} , and $\boldsymbol{\sigma}$ directly used in sampling, and
2. set the auxiliary strategy-specific parameters (cumulation paths, best-so-far solution, or other statistics describing the past evolution).

Again, several distinctive features can be observed:

Feature 3. *Is the model re-estimated from scratch each generation?*

The model parameters are set *only* on the basis of the individuals in the current population (like in ML-EDA).

Feature 4. *Does the model building phase use selected and/or discarded individuals?*

The discarded individuals are used in only a few works even though they offer various possibilities.

Feature 5. *How to determine the model center for the next generation?*

Answer to this question amounts to defining the equation for setting $\boldsymbol{\mu}^{(g)}$. The next sample can be centered around the best individual in current population, around the best-so-far individual, around the mean of the selected individuals (ML-EDA), around the weighted mean of the best individuals (CMA-ES), around a place where we anticipate that the mean should move (AMS), etc.

Feature 6. *How much should the distribution be enlarged?*

In other words, what should the global step-size setting be? The global step-size c can be 1 (ML-EDA), a constant (VS), or it can be adapted (AVS), or eventually used only sometimes (CT, SDR).

¹ These isotropic distributions are sampled so that (1) a direction vector is selected uniformly by selecting a point on hypersphere, and (2) this direction vector is multiplied by a radius sampled from 1D Gaussian or Cauchy distribution, respectively.

Feature 7. What should the shape of the distribution be?

The answer lies in the way of computing the rotation matrix \mathbf{R} and scaling factors σ . These are usually closely related. They can be set e.g. by eigendecomposition of the covariance matrix of the selected data points (ML-EDA). In that case the σ is a vector of standard deviations in the main axes and \mathbf{R} is a matrix of vectors pointing in directions of the main axes. AE offers an alternative way of estimating the σ and \mathbf{R} .

Feature 8. What should the reference point² be?

Closely related to the previous feature, it has a crucial impact on the algorithm behaviour. If we take the selected data points X_S , subtract their mean \bar{X}_S , and perform the eigendecomposition

$$[\sigma^2, \mathbf{R}] \leftarrow \text{eig}(X_S, \bar{X}_S), \text{ where } \text{eig}(X, x_r) \stackrel{\text{def}}{=} \text{eig}\left(\frac{1}{N-1}(X - x_r)(X - x_r)^T\right)$$

we arrive at the approach ML-EDAs are using. If we change the reference point x_r from \bar{X}_S to $\mu^{(g-1)}$ (so that $[\sigma^2, \mathbf{R}] \leftarrow \text{eig}(X_S, \mu^{(g-1)})$), then we get the principle behind CMA-ES—we estimate the distribution of selected mutation steps.

4 New Possibilities for SLS

Since many of the features are independent of each other, it makes sense to create new algorithms by combining previously unexplored combinations, and asses the quality of the newly constructed algorithms. Due to the space limitations, let us very briefly discuss only some possibilities for the reference point of the distribution shape estimate (feature 8, F8) if we allow the model building phase to use the selected as well as discarded solutions (F4). In the following, the Gaussian distribution is used (F1) in all generations (F2). The model is re-estimated from scratch each generation with the exception of CMA-ES-like configuration where $\mu^{(g-1)}$ is used as the reference point (F3). A conservative setting was chosen for the distribution center in this article: \bar{X}_S is used similarly to ML-EDA (F5). The distribution is not enlarged, $c = 1$ (F6), and the shape is estimated by eigendecomposition of XX^T matrix of certain vectors in X (F7).

Interesting configurations can be envisioned in setting the reference point for estimation of the distribution shape. Let \bar{X}_B and \bar{X}_W be (possibly weighted) averages of several best selected and several best discarded solutions in the current population, respectively. X_S and X_D are solutions selected and discarded, respectively, by the selection operator. Furthermore, $X_B \subset X_S$ and $X_W \subset X_D$. Instead of using $\text{eig}(X_S, \bar{X}_S)$ (which is ML-EDA and converges prematurely, see Fig. 1, upper left) or $\text{eig}(X_S, \mu^{(g-1)})$ (which is successful CMA-ES-like approach, see Fig. 1, upper right), we can use $\text{eig}(X_S, \bar{X}_B)$ (see Fig. 1, lower left). Compared to ML-EDA, this might result in greater spread in the gradient direction on the slope, but as a whole the estimates are still too low and the algorithm converges prematurely. In the neighborhood of the optimum, the MLE is recovered since \bar{X}_B is expected to be the same as \bar{X}_S .

² Note the difference between the model center (see feature 5) and the reference point. We can e.g. estimate the shape of the distribution based on selected points when the worst point is taken as the reference, and then center the learned distribution around the best point.

Another option is to use $\text{eig}(X_W, \bar{X}_B)$ (see Fig. 1, lower right). As can be seen, it might give the algorithm additional burst, since it located the optimum after 50 generations which is not the case for any other configuration.

It is, of course, impossible to draw some far-reaching conclusions based on the presented pictures. Statistical analysis on broad class of problems and dimensionalities is needed and it is questionable if some of these methods can beat the finely tuned CMA-ES. In general, however, there are many choices better than the ML-EDA approach.

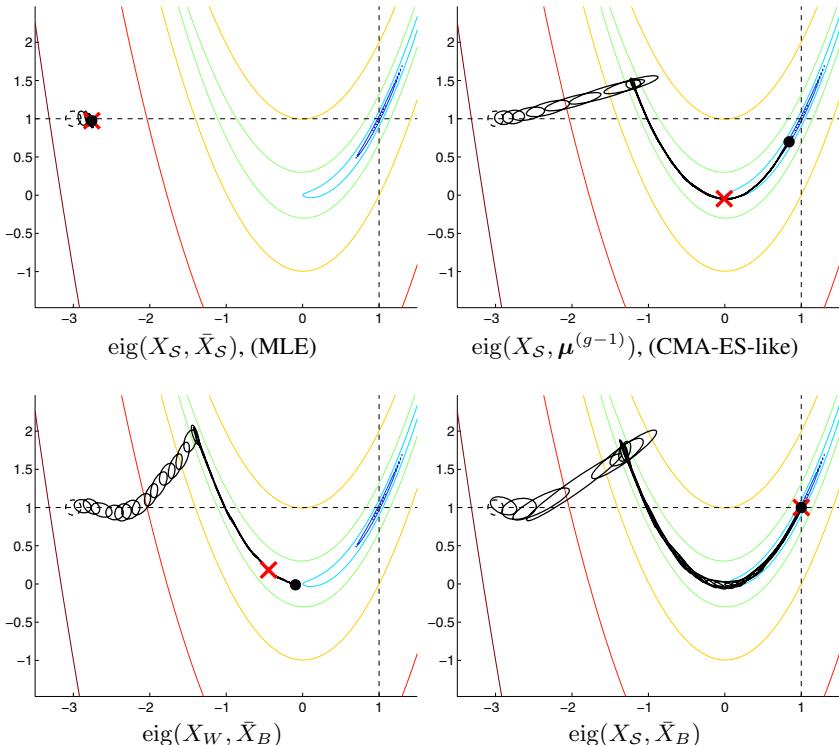


Fig. 1. SLS with various approaches of estimating the shape of the distribution on the 2D Rosenbrock function. Red cross: $\mu^{(50)}$, black dot: $\mu^{(100)}$. Initialization: $\mu^{(0)} = (-3, 1)$, $\sigma^{(0)} = (0.1, 0.1)$. No enlargement of the estimated shape takes place, $c = 1$. Population size 200 and truncation selection with selection proportion $\tau = 0.3$ was used for all pictures.

5 Conclusions

This paper surveyed recent contributions in the area of SLS techniques using single-peak search distribution. A broad set of methods and tweaks exist in this field—various similarities and differences were pointed out. Based on the lessons learned from these methods, a set of rather independent features was compiled; these features can be used as a taxonomy to classify various SLS techniques. The taxonomy offers also many previously unexplored feature combinations that can result in potentially successful algorithms. Exploring these various possibilities remains as a future work.

Acknowledgements

The author is supported by the Grant Agency of the Czech Republic with the grant no. 102/08/P094 entitled “Machine learning methods for solution construction in evolutionary algorithms”.

References

1. Hoos, H.H., Stützle, T.: *Stochastic Local Search: Foundations & Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Francisco (2004)
2. Voss, S.: Book review: H. H. Hoos and T. Stützle: Stochastic local search: foundations and applications (2005). *Mathematical Methods of Operations Research* 63(1), 193–194 (2006)
3. Beyer, H.G., Schwefel, H.P.: Evolution strategies – a comprehensive introduction. *Natural Computing* 1(1), 3–52 (2002)
4. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading (1989)
5. Rechenberg, I.: Evolutionsstrategien. In: Schneider (ed.) *Simulationsmethoden in der Medizin und Biologie*, pp. 83–113. Springer, Berlin (1978)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
7. Larrañaga, P., Lozano, J.A. (eds.): *Estimation of Distribution Algorithms*. GENA. Kluwer Academic Publishers, Dordrecht (2002)
8. Rudlof, S., Köppen, M.: Stochastic hill climbing by vectors of normal distributions. In: First Online Workshop on Soft Computing, Nagoya, Japan (1996)
9. Larrañaga, P., Etxeberria, R., Lozano, J.A., Sierra, B., Inza, I., Peña, J.M.: A review of the cooperation between evolutionary computation and probabilistic graphical models. In: Rodriguez, A.A.O., Ortiz, M.R.S., Hermida, R.S. (eds.) *CIMAF 1999, Second Symposium on Artificial Intelligence, La Habana. Adaptive Systems*, pp. 314–324 (1999)
10. Larrañaga, P., Lozano, J.A., Bengoetxea, E.: Estimation of distribution algorithms based on multivariate normal distributions and Gaussian networks. Technical Report KZZA-IK-1-01, Dept. of Computer Science and Artificial Intelligence, University of Basque Country (2001)
11. Bosman, P.A., Thierens, D.: Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In: *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pp. 197–200 (2000)
12. Bosman, P.A.N., Thierens, D.: Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) *PPSN 2000. LNCS*, vol. 1917, pp. 767–776. Springer, Heidelberg (2000)
13. Yuan, B., Gallagher, M.: On the importance of diversity maintenance in estimation of distribution algorithms. In: Beyer, H.G., O'Reilly, U.M. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2005*, vol. 1, pp. 719–726. ACM Press, New York (2005)
14. Ocenasek, J., Kern, S., Hansen, N., Koumoutsakos, P.: A mixed bayesian optimization algorithm with variance adaptation. In: Yao, X. (ed.) *PPSN 2004. LNCS*, vol. 3242, pp. 352–361. Springer, Heidelberg (2004)
15. Grahl, J., Minner, S., Rothlauf, F.: Behaviour of UMDAc with truncation selection on monotonous functions. In: *IEEE Congress on Evolutionary Computation, CEC 2005*, vol. 3, pp. 2553–2559 (2005)

16. Gonzales, C., Lozano, J.A., Larrañaga, P.: Mathematical modelling of UMDAc algorithm with tournament selection. *International Journal of Approximate Reasoning* 31(3), 313–340 (2002)
17. Grahl, J., Bosman, P.A.N., Rothlauf, F.: The correlation-triggered adaptive variance scaling IDEA. In: Proceedings of the 8th annual conference on Genetic and Evolutionary Computation Conference – GECCO 2006, pp. 397–404. ACM Press, New York (2006)
18. Bosman, P.A.N., Grahl, J., Rothlauf, F.: SDR: A better trigger for adaptive variance scaling in normal EDAs. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation, pp. 492–499. ACM Press, New York (2007)
19. Pošík, P.: Preventing premature convergence in a simple EDA via global step size setting. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 549–558. Springer, Heidelberg (2008)
20. Pošík, P., Franc, V.: Estimation of fitness landscape contours in EAs. In: Genetic and evolutionary computation conference – GECCO 2007, New York, NY, USA, pp. 562–569. ACM Press, New York (2007)
21. Bosman, P., Grahl, J., Thierens, D.: Enhancing the performance of maximum-likelihood Gaussian EDAs using anticipated mean shift. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 133–143. Springer, Heidelberg (2008)
22. Jastrebski, G.A., Arnold, D.V.: Improving evolution strategies through active covariance matrix adaptation. In: IEEE Congress on Evolutionary Computation – CEC 2006, pp. 2814–2821 (2006)
23. Arnold, D.V., Van Wart, D.C.S.: Cumulative step length adaptation for evolution strategies using negative recombination weights. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 545–554. Springer, Heidelberg (2008)
24. Kern, S., Müller, S.D., Hansen, N., Büche, D., Očenášek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms— a comparative review. *Natural Computing* 3(1), 77–112 (2004)
25. Pošík, P.: Truncation selection and gaussian EDA: Bounds for sustainable progress in high-dimensional spaces. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 525–534. Springer, Heidelberg (2008)
26. Yao, X., Liu, Y.: Fast evolution strategies. *Control and Cybernetics* 26, 467–496 (1997)
27. Obuchowicz, A.: Multidimensional mutations in evolutionary algorithms based on real-valued representation. *Int. J. Systems Science* 34(7), 469–483 (2003)
28. Hansen, N., Gempeler, F., Auger, A., Koumoutsakos, P.: When do heavy-tail distributions help? In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 62–71. Springer, Heidelberg (2006)
29. Miqéléz, T., Bengoetxea, E., Mendiburu, A., Larrañaga, P.: Combining Bayesian classifiers and estimation of distribution algorithms for optimization in continuous domains. *Connection Science* 19(4), 297–319 (2007)
30. Wojtusiak, J., Michalski, R.S.: The LEM3 system for non-darwinian evolutionary computation and its application to complex function optimization. Reports of the Machine Learning and Inference Laboratory MLI 04-1, George Mason University, Fairfax, VA (February 2006)
31. Hansen, N.: Adaptive encoding: How to render search coordinate system invariant. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 205–214. Springer, Heidelberg (2008)
32. Pošík, P.: On the Use of Probabilistic Models and Coordinate Transforms in Real-Valued Evolutionary Algorithms. PhD thesis, Czech Technical University in Prague, Prague, Czech Republic (2007)

Evolutionary Optimization Guided by Entropy-Based Discretization

Guleng Sheri and David W. Corne

Department of Computer Science, Heriot-Watt University, Edinburgh, UK
gls3@macs.hw.ac.uk, dwcorne@gmail.com

Abstract. The Learnable Evolution Model (LEM) involves alternating periods of optimization and learning, performing extremely well on a range of problems, a specialises in achieving good results in relatively few function evaluations. LEM implementations tend to use sophisticated learning strategies. Here we continue an exploration of alternative and simpler learning strategies, and try Entropy-based Discretization (ED), whereby, for each parameter in the search space, we infer from recent evaluated samples what seems to be a ‘good’ interval. We find that LEM(ED) provides significant advantages in both solution speed and quality over the unadorned evolutionary algorithm, and is usually superior to CMA-ES when the number of evaluations is limited. It is interesting to see such improvement gained from an easily-implemented approach. LEM(ED) can be tentatively recommended for trial on problems where good results are needed in relatively few fitness evaluations, while it is open to several routes of extension and further sophistication. Finally, results reported here are not based on a modern function optimization suite, but ongoing work confirms that our findings remain valid for non-separable functions.

1 Introduction

The Learnable Evolution Model (LEM) [13] is a general hybrid approach, in which the overall idea is to combine evolutionary search and learning, where ‘evolution’ periods are informed by previous ‘learning’ periods. E.g., following n generations of an evolutionary algorithm (EA), we might then apply learning (perhaps a neural network, or a decision tree learner) which will exploit the information gained so far by trying to learn to predict fitness (or categories of fitness, such as ‘good’ and ‘bad’) from vectors of gene values. The result of the learning phase is then used somehow to inform the next period of evolution. The way in which learning influences evolution is not restricted. For example, the learned mapping from gene values to fitness could be used to predict the fitness of children before they are evaluated, and the evolution phase then discards, without evaluation, children that are predicted to be of poor fitness. Alternatively the learned model may be used to constrain the genetic operators in such a way that children are more likely to be fit. Or, the learned model may be used to ‘repair’ children, and so on.

In [13], the learning method was AQ15 [11,16], and the results of this, along with the results emerging from continuing development of LEM from Michalski’s

group [17,18], were highly promising. In particular LEM led to dramatic speedup on a range of function optimization problems as well as on a real-world problem. In other work inspired by LEM, a multiobjective version (using C4.5 as the learner), significantly accelerated and improved solution quality for large-scale problems in water distribution networks [10].

Of course, the essential idea of combining learning and evolution is not restricted to LEM. While LEM emerged from the machine learning community, Estimation of Distribution Algorithms (EDAs) blossomed in the evolutionary computation community [12]. Both LEM and EDA now have several published variants (particularly EDA variants), and it is interesting to contrast the two. While EDAs focus on modelling as the key force behind search (i.e. search is guided closely by the modelling, with new sample points in the space generated directly from the model), in LEM the evolutionary component is most responsible for the search (i.e. new points are sampled mainly in the familiar way by using genetic operators on a population of chromosomes), with guidance from learning processes. Most interestingly, recent results compare EDAs and LEM3 directly [18], using a variety of EDA implementations [5]. Comparisons showed LEM3 (with AQ) from 15–230 times faster than EMNA_{global}, and always achieving a superior final solution. Also, recent years have seen the appearance of *hybrids* of EDAs and other search methods [19,14,20]. When contrasting LEM with EDA, it is sensible to say that LEM is similar to a hybrid of an EDA and an EA; this seems consonant with the success that has so far been reported for EDA/EA hybrids.

Clearly, more research is warranted towards understanding the design and application of algorithms within the LEM framework. In previous work [4], the authors explored how well a LEM algorithm performs when the learning technique is perhaps the simplest possible such mechanism: *k*-nearest neighbour. In LEM(KNN), the learning phase comprised only the process of identifying the best and worst 30% of the current population; subsequent evolution only allowed evaluation of children for whom the majority of their 5 nearest neighbours from these sets were in the higher-fitness group. Although not able to produce results that rivalled those of LEM(AQ) [13], this simple intervention of learning provided very significant speedup and solution quality improvements over the unchanged EA. In this paper the investigation of alternative learning methods is continued by the use of a quite different approach. We use the concept of *entropy-based discretization* (ED), whereby, for each parameter in the search space, we infer from recent evaluated samples what seems to be a ‘good’ interval for that parameter. Subsequent periods of evolutionary search are biased towards exploring these ‘good’ intervals. Over a range of function optimization problems, we find that the resulting LEM(ED) methods again provide significant advantages in both solution speed and quality over the unadorned evolutionary algorithm, and on most problems it is also superior to CMA-ES when the number of evaluations is limited. Though details are not included here, LEM(ED) is also significantly better than LEM(KNN).

In the remainder we continue as follows. Section 2 provides more detail on LEM and on our LEM(ED) variation(s). Section 3 covers experiments and results on some test problems, and we conclude and discuss in section 4.

2 LEM and LEM(ED)

2.1 The LEM Framework

In LEM(AQ) [13], an initial population is divided into high-performance (H-group) and low-performance (L-group) groups according to fitness, to be used as positive and negative training examples for AQ learning. The outcome of learning is a set of rules which predict a class label (i.e. H-group or L-group) for a chromosome. LEM(AQ) then proceeds with an otherwise normal EA, except that the operators generate new individuals only with gene values within the ranges sanctioned by the recently learned rules. LEM(AQ) then continues for a specified amount of generations, and then pauses for more learning based on the current population. This feeds into the next stage of evolution, and so on. LEM(AQ) has many additional details that mediate the transitions between learning and evolution, and we refer readers to [13] for more details.

LEM(AQ), thus overviewed, is simply an instantiation of the wider LEM framework, which allows for creativity in the choices of learning method, and the way in which learning and evolution interact. Here, we continue an investigation of simple instantiations of this framework. The key details in the instantiation we use in this paper are as follows: the learning method is entropy-based discretization (described next); we tightly interleave learning and exploration in the initial generations, and then we cease further learning entirely; finally, the way in which learning affects evolution, similarly to the LEM(AQ) case described above, is that we constrain the generation of new chromosomes according to the learned model.

2.2 Entropy-Based Discretization(ED)

We use entropy-based supervised binary discretization [1], taken from the machine learning literature. As is well-known, for example, ID3[2] and C4.5[3] use entropy measures to guide the development of a decision tree. In particular, in data mining there is often a need to discretize the range of a continuous parameter (into, for example, two intervals labelled ‘low’ and ‘high’). This is done by considering all possible cutpoints for the boundary between the intervals, and calculating an entropy measure for those cutpoints. In the case of data mining, entropy is based on the way in which the target class values are distributed over the two intervals, and the cutpoint that provides the best entropy measure is chosen. In our usage of ED, we basically do the same, but considering ‘good fitness’ and ‘bad fitness’ to be the target class values. I.e. we choose a cutpoint that seems to best separate ‘good’ from ‘bad’ in the current population.

2.3 The LEM(ED) Algorithm

As with LEM(AQ), LEM(ED) divides the *current population* into high-performance (H-group) and low-performance (L-group) groups according to their fitness values and a given *threshold* (say, 30% – that is, the fittest 30% form the H-group and the worst 30% form the L-group). This is then saved as the *learning population*. Individuals of the H-group and L-group in the *learning population* form the training examples used by ED. ED is applied to each parameter i independently; the output of ED for parameter i is the pair of intervals $[min_i, cut_point_i]$ and $[cut_point_i, max_i]$ where cut_point_i is the splitting point with minimum entropy, and min_i and max_i are the lower and upper limits specified for parameter i .

The two intervals in each such pair are then labelled *good* and *bad* (the good interval is the one with the highest proportion of points from the H-group). Figure 1 illustrates this. On the left, we show the cutpoint (and hence division into good and bad intervals) that might be learned from a population of fitnesses for a given parameter. In the case on the left, the good interval happens to contain the value of that parameter that appears in the global optimum; in the case on the right, the optimum is missed by the good interval. The idea of LEM(ED) is to guide evolution by biassing further search within the ‘good’ intervals, and this is, of course, appropriate if the case on the left occurs more often than the case on the right, or if not, we should allow enough exploration and adaption to ‘recover’ from mislabelled intervals. We have done various investigations around this issue and expect good adaptive schemes to emerge in future, but in the present work we consider only two basic variants of LEM(ED). In each one, learning of intervals ceases after a generation that has seen no improvement in the best solution. In LEM(ED)1, further generations continue with the underlying EA (unconstrained), and in LEM(ED)2, further generations continue with the underlying EA, but biassed by the last learned intervals.

When the intervals for all parameters are labelled, LEM(ED) begins the instantiation procedure. New individuals for the next generation are now generated according the *good* intervals. In the current work, we achieve this by generating

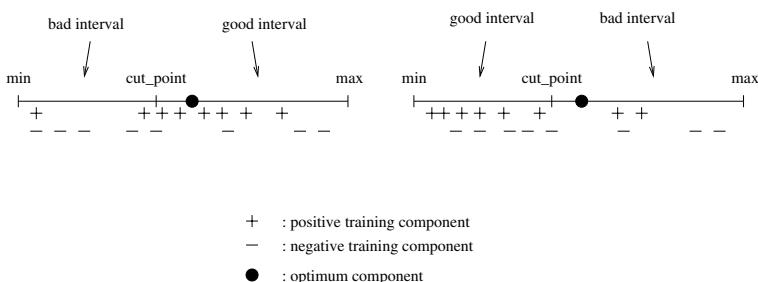


Fig. 1. Illustrating entropy-based discretization, showing correctly (left) and incorrectly (right) labelled interval pairs

new individuals uniformly at random from one of the intervals, using a high probability to choose the good interval.

When the new population is created, ED is applied again to the current population, and the process of ED and generation of new individuals until the latest instantiation step leads to no improvement in best fitness. When the learning mode is finished, we continue with an ordinary EA. In the case of LEM(ED)1, the EA benefits only from a good (hopefully) initial population emerging from the learning phase just described. In LEM(ED)2, the EA additionally uses the final set of learned intervals, biassing the generation of new individuals.

‘Overview’ pseudo-code for LEM(ED) is as follows:

1. Set values for *population size*, mutation probability, crossover probability, *learning threshold* and set elite-preserve operator option.
2. Generate initial population: Choose a method to create the initial population with *population size* and evaluate this population.
3. Begin learning mode :
 - (a) Derive extrema: Copy the *current population* as the *learning population*, from which create the high fitness group (H-group) and low fitness group (L-group) according to fitness values and *threshold*. These two groups are stored as positive and negative training data for learning algorithm.
 - (b) Apply ED: For each parameter, consider each value as the potential cut-point and calculate the information gain for each such point, choosing the best as the *cut_point* for this parameter. The output of this step on each dimension is two intervals having the form $\langle \min, \text{cut_point} \rangle$, and $\langle \text{cut_point}, \max \rangle$.
 - (c) Label the learned intervals: For the output intervals on each parameter, label them as *good* and *bad* intervals by observing the relative proportions of gene values from the H-group and L-group.
 - (d) Instantiate new individuals: After the discretization and labelling procedures, the new individuals for next generation are generated from the good intervals in each dimension.
 - (e) evaluate each of the new individuals; if there has been no improvement in best fitness, terminate learning and switch to EA mode.
 - (f) Update H-group and L-group: When the new population is generated, the H-group and L-group are recalculated. Return to step (b).
4. Begin evolution mode: The EA can of course be freely designed, and may or may not use the intervals generated at the end of the learning phase.

LEM(ED) will terminate if the optimum (if known) is reached, or the maximum allowed number of generations is reached. In LEM(ED)1, the evolution phase runs a straightforward EA, described next, whose initial population is the output of the learning phase. In LEM(ED)2, the EA is the same, except that mutation uses the final set of intervals learned in the learning phase; mutations to parameter i will have a high chance of choosing a value from the ‘good’ interval for i .

3 Experiments and Results

We tested LEM(ED)1 and LEM(ED)2 by running them on a set of functions that were used in the original LEM paper [13], augmented to provide a more thorough test. We compared with the underlying EA (i.e. LEM(ED) without the learning phase), and, to achieve an understanding in relation to state of the art performance, we chose to compare also with CMA-ES[7,8].

In all cases the encoding was a vector of real-valued genes within the specified interval. The EA used binary tournament selection [9], elitism, uniform crossover [15] with probability 0.5, and uniform mutation with probability $1/(length\ of\ chromosome)$.

In LEM(ED) the *learning threshold* is 0.3, the instantiation method is implemented according to a probability; 80% new individuals are generated from the *good* interval, 20% are from the *bad* interval. In LEM(ED)2's evolution mode, the mutation is implemented with a normal distribution whose mean is centred over the good interval for the parameter in question, and with variance 1.

The main parameters for CMAES(μ, λ) are μ , number of parent individuals, λ , the number of offspring, and the initial standard deviations σ . Here, we implement CMAES(50, 100) and *sigma* is set to one third of the range of each variable. Finally, EA and LEM(ED) population size was 100.

For completeness, details of the test functions used are:

a) Rastrigin's function:

$$f(x_1, \dots, x_n) = 3.0n + \sum_{i=1}^n (x_i^2 - 3.0 \cos(2\pi x_i)) \quad (1)$$

where $n = 30$ and $-5.12 \leq x_i \leq 5.12$.

b) Griewangk's function:

$$f(x_1, \dots, x_n) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (2)$$

where $n = 30$ and $-600 \leq x_i \leq 600$.

c) Rosenbrock's function:

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (3)$$

where $n = 30$ and $-2.048 \leq x_i \leq 2.048$.

d) Ackley's function:

$$f(x_1, \dots, x_n) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) \quad (4)$$

where $n = 30$ and $-30 \leq x_i \leq 30$.

e) de Jong's function 3:

$$f(x_1, \dots, x_n) = \sum_{i=1}^n \text{integer}(x_i) \quad (5)$$

where $n = 30$ and $-5.12 \leq x_i \leq 5.12$.

f) de Jong's function 4:

$$f(x_1, \dots, x_n) = \sum_{i=1}^n ix_i^4 + \text{Gauss}(0, 1) \quad (6)$$

where $n = 30$ and $-1.28 \leq x_i \leq 1.28$.

g) Schwefel's function:

$$f(x_1, \dots, x_n) = 418.9829n + \sum_{i=1}^n x_i \sin(\sqrt{|x_i|}) \quad (7)$$

where $n = 30$ and $-500 \leq x_i \leq 500$.

Table 1 summarises the results of 100 runs of each algorithm on each function, with means and standard deviations recorded at 10, 20, 50 and 100 generations (multiply by 100 for number of fitness evaluations). Table 1 also provides information about the mean generation number at which learning was automatically terminated in the LEM(ED) variants. In interpreting the results, note that de Jong function 3 is a maximisation problem, and all others are minimization.

In the below, statements of significance are based on randomisation tests ([6]), and made only when confidence was above 99%. Inspection of standard deviations also clearly supports the statements made. With only one exception (LEM(ED)2 on Schwefel at 50 and 100 generations) the LEM(ED) variants are always significantly superior to the underlying EA. To some extent, the underlying EA can be seen as a 'straw man', and it is used here only as a baseline, with improvement to be expected. However, since LEM(ED)2 biasses itself strongly to learned intervals, it is quite possible that some functions can lead to it being deceived and misled. It is interesting and promising that this usually does not seem to happen, however LEM(ED)2's early fast progress on the Schwefel function clearly leads it in the wrong direction. This situation is the same for CMAE-ES, which is always beaten by both the EA and LEM(ED)1 on Schwefel, and beaten also by LEM(ED)2 until the 100-generations point.

Notwithstanding the case of the Schwefel function, the CMA-ES comparisons provide a much more exacting test for LEM(ED). With the exception of de Jong's function 3, one of the LEM(ED) variants is always superior to CMA-ES (and the GA) at the 1,000 and 2,000 evaluation points. At the 5,000 evaluations point, a LEM(ED) variant outperforms CMA-ES on de Jong's function 4, Rastrigin, Griewangk and Schwefel, and at 10,000 evaluations this reduces to Rastrigin and Schwefel, with ties for the de Jong functions. In general, CMA-ES, which is itself a sophisticated hybrid of learning and evolution, overtakes LEM(ED) (and the

Table 1. Summarising 100 independent runs of each algorithm on each problem: means and standard deviations after 10, 20, 50 and 100 generations (1,000, 2,000, 5,000 and 10,000 evaluations); in the first column of the final (100 generations) section, the number in brackets after the function name indicates the mean generation number at which learning terminated in the two LEM(ED) variants

Functions	EA	LEM(ED)1	LEM(ED)2	CMAES
10 generations				
Dejong3	84.1(0.89)	135.4(0.62)	136.1(0.01)	150(0)
Dejong4	26.7(1.3)	1.68 (0.18)	1.96 (0.05)	50.22 (8.2)
Rastrigin	148.52(1.1)	92.91(1.8)	91.98 (0.16)	196.1(0.92)
Griewangk	243.5(0.25)	64.21 (0.89)	64.79 (1.4)	256.8 (2.9)
Rosenbrock	2105.4 (28)	397.6 (5.4)	398.3 (11)	3663.7 (150)
Ackley	18.46 (0.043)	13.67 (0.081)	13.67(0.082)	20.14(0.021)
Schwefel	6733.6 (36)	5542.73 (82)	5196.01 (24)	8400.76 (510)
20 generations				
Dejong3	111.08 (1.2)	139.76(0.73)	142.72 (0.3)	150 (0)
Dejong4	8.732 (0.26)	-0.910 (0.052))	-0.820 (0.030)	3.657(0.061)
Rastrigin	98.77(1)	58.31(0.21)	56.77 (1.9)	107.45 (2.6)
Griewangk	116.19(1.5)	7.61 (0.058)	7.93(0.25)	187.45(6.5)
Rosenbrock	942.47 (12)	112.13(5.1)	110.096 (1.8)	664.91 (32)
Ackley	16.068(0.04)	6.94 (0.17)	6.90 (0.062)	16.48 (0.049)
Schwefel	4607.46 (87)	3686.41 (9.3)	3693.12 (8.0)	8016.14 (450)
50 generations				
Dejong3	134.76(0.13)	146.5 (0.087)	149.93 (0.012)	150(0)
Dejong4	0.298(0.19)	-2.176 (0.041)	-2.192 (0.068)	0.0012(0.00016)
Rastrigin	50.9(1.1)	18.13(0.53)	14.53 (0.63)	62.01 (0.5)
Griewangk	36.70(3.1)	2.31(0.15)	2.58 (0.14)	3.399 (0.16)
4 Rosenbrock	331.65 (8.6)	81.79 (3.9)	79.97 (0.95)	40.1 (0.71)
Ackley	11.63(0.13)	3.57(0.11)	3.49(0.14)	3.054 (0.033)
Schwefel	2205.78 (74)	1795.15 (83)	3258.36 (3.7)	6637.8 (820)
100 generations				
Dejong3 (5)	145.41 (0.42)	149.46(0.08)	150 (0)	150 (0)
Dejong4 (11)	-1.19 (0.054)	-2.64(0.1)	-2.71 (0.054)	1.9e-5(2e-6)
Rastrigin (40)	25.29 (0.84)	9.13 (0.07)	4.76 (0.09)	36.09 (0.22)
Griewangk (58)	11.94(0.3)	1.39 (0.005)	1.98(0.11)	0.38 (0.079)
Rosenbrock (47)	178.3 (6.4)	55.39 (1.9)	47.7 (0.25)	28.01 (0.014)
Ackley (50)	8.061 (0.05)	2.43 (0.16)	0.701(0.069)	0.1694 (0.00074)
Schwefel (2)	1008.87 (42)	827.694 (12)	2871.39 (3.2)	2621.92 (1200)

vast majority of other algorithms) as we consume more function evaluations. Regarding the LEM(ED) design tested here, this is not surprising since our LEM(ED) variants use a single learning phase followed by an EA, while CMA-ES is continually learning and adapting. LEM(ED)'s performance in the 1,000–5,000 evaluations regime is nevertheless encouraging, and there may be considerable value in more sophisticated adaptive versions.

4 Concluding Discussion

Our choice of function suite follows those used in the original LEM publications. However, such suites are now superseded by those described in the CEC 2005 Challenge [21], which emphasises non-separability and other measures that are likely to make functions difficult. Since most of the functions tested herein are, however, separable, the criticism can be made that the findings may well not generalise to nonseparable functions. However, following preliminary and ongoing work we can confirm that the LEM(ED) variants here show entirely similar relative (to basic GA and to CMA-ES) performance properties as found here, and this will be the topic of further dissemination.

So, on the functions studied herein, we find that LEM(ED), a method which incorporates a simple entropy-based discretization method in the initial part of a EA run, clearly outperforms its EA component alone, and also generally outperforms CMA-ES during the initial several-thousand fitness evaluations. Also, though we omit details here, LEM(ED) outperforms LEM(KNN) [4]. This adds to evidence that even straightforward learning mechanisms provide considerable benefit to an EA, especially for accelerating the search.

Lines of further work that seem warranted include testing on a more accepted set of optimization challenge functions, and investigating repeated phases of ED-based learning (rather than a single phase at the beginning). Our investigations so far have focused on tightly coupling ED and instantiation, which (as we find in preliminary experiments) is best limited, rather than continued throughout the run, otherwise the learned intervals can be deceived and results suffer. However we are yet to investigate (which would be highly suited to the LEM framework) the interleaving of further ED/instantiation phases with phases of several generations of evolution. Meanwhile, the information inherent in the learned intervals could be used more creatively in later phases, in various ways. Also, a more sophisticated termination criterion for the ED phase would be beneficial, since we note that the ‘better’ sets of intervals are often those learned a handful of generations before the cessation of fitness improvement.

More generally, more study seems warranted in the area of learning/evolution combinations (both in terms of LEM-framework instantiations, and also in terms of organising the knowledge in this important area that is currently spread widely in the literature).

References

1. Hussain, F., Liu, H., Tan, C.L., Dash, M.: Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery* 6(4), 393–423 (2002)
2. Quinlan, J.R.: Induction of decision trees. *Machine Learning* 1, 81–106 (1986)
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
4. Sheri, G., Corne, D.: The Simplest Evolution/Learning Hybrid: LEM with KNN. In: Proc. IEEE CEC 2008, pp. 3244–3251 (2008)

5. Bengoetxea, E., Miquelez, M., Larranga, P., Lozano, J.A.: Experimental results in function optimization with EDAs in Continuous Domain. In: [12] (2002)
6. Edgington, E.S.: Randomization tests, 3rd edn. Marcel-Dekker, New York (1995)
7. Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size. In: Proc. IEEE CEC 2005, pp. 1769–1776 (2005)
8. Hansen, N.: The CMA Evolution Strategy: A Comparing Review. In: Lozano, J.A., et al. (eds.) Towards a new evolutionary computation. Advances in estimation of distribution algorithms, pp. 75–102. Springer, Heidelberg (2006)
9. Goldberg, D.E.: Genetic Algorithms in Search. Optimization and Machine Learning. Addison-Wesley, Reading (1989)
10. Jourdan, L., Corne, D., Savic, D., Walters, G.: Hybridising rule induction and multiobjective evolutionary search for optimizing water distribution systems. In: Proc. of the 4th Hybrid Intelligent Systems conference, pp. 434–439. IEEE Computer Society Press, Los Alamitos (2005)
11. Kaufmann, K., Michalski, R.S.: Learning from inconsistent and noisy data. In: The AQ18 approach, 11th Int'l. Symp. on Foundations of Intelligent Systems (1999)
12. Larranaga, P., Lozano, J.A. (eds.): Stimulation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)
13. Michalski, R.S.: Learnable Evolution Model Evolutionary Processes Guided by Machine Learning. Machine Learning 38, 9–40 (2000)
14. Pena, J.M., Robles, V., Larranaga, P., Herves, V., Rosales, F., Perez, M.S.: GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 361–371. Springer, Heidelberg (2004)
15. Syswerda, G.: Uniform Crossover in Genetic Algorithms. In: Proc. of 3rd International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., San Francisco (1989)
16. Wnek, J., Kaufmann, K., Bloedorn, E., Michalski, R.S.: Inductive Learning System AQ15c: The method and user's guide. Reports of the Machine Learning and Inference Laboratory, MLI95-4, George Mason University, Fairfax, VA, USA (1995)
17. Wojtusiak, J., Michalski, R.S.: The LEM3 System for Non-Darwinian Evolutionary Computation and Its Application to Complex Function Optimization, Reports of the Machine Learning and Inference Laboratory, MLI 05-2, George Mason University, Fairfax, VA, USA (2005)
18. Wojtusiak, J., Michalski, R.S.: The LEM3 implementation of learnable evolution model and its testing on complex function optimization problems. In: Proc. GECCO 2006 (2006)
19. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Hybrid estimation of distribution algorithm for global optimisation. Engineering Computations 21(1), 91–107 (2003)
20. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Estimation of distribution algorithm with 2-opt Local Search for the Quadratic Assignment Problem. In: Lozana, Larranaga, Inza, Bengoetxea (eds.) Towards a new evolutionary computation: Advances in estimation of distribution algorithms (2006)
21. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, A.: Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization. Technical Report 2005005, Nanyang Technological University, Singapore (2005)

The Influence of Population and Memory Sizes on the Evolutionary Algorithm's Performance for Dynamic Environments

Anabela Simões^{1,2} and Ernesto Costa²

¹ Department of Informatics and Systems Engineering, Coimbra Polytechnic

² Centre for Informatics and Systems of the University of Coimbra

abs@isec.pt, ernesto@dei.uc.pt

Abstract. Usually, evolutionary algorithms keep the size of the population fixed. In the context of dynamic environments, many approaches divide the main population into two, one part that evolves as usual another that plays the role of memory of past good solutions. The size of these two populations is often chosen off-line. Usually memory size is chosen as a small percentage of population size, but this decision can be a strong weakness in algorithms dealing with dynamic environments. In this work we do an experimental study about the importance of this parameter for the algorithm's performance. Results show that tuning the population and memory sizes is not an easy task and the impact of that choice on the algorithm's performance is significant. Using an algorithm that dynamically adjusts the population and memory sizes outperforms standard approach.

1 Introduction

Evolutionary Algorithms (EAs) have been used with success in a wide area of applications, involving environments either static or dynamic. Traditional EAs usually have a number of control parameters that are specified before starting the algorithm. These parameters include, for example, the probabilities of mutation and crossover or the population size. The effects of setting the parameters of EAs has been the subject of extensive research and in the last years several approaches using self-adaptive EAs, which can adjust the parameters on-the-fly, have been proposed. The study of EAs for *stationary domains* has previously focused on the adjustment of parameters of genetic operators. The choice of the size of the population has received less attention by researchers. However, if we look to natural systems that inspire EAs, the number of individuals of a certain species changes over time and tends to become stable around appropriate values, according to environmental characteristics or natural resources ([1]). In evolutionary computation the population size is usually an unchanging parameter and finding an appropriate dimension is a difficult task. Several adaptive population sizing methods have been suggested (see [2] for a review). When EAs are used to deal with *dynamic environments*, some modifications have to be introduced to avoid premature convergence since convergence is disadvantageous

when a change happens. These enhancements include increasing diversity after a change ([3]), maintaining diversity throughout the run ([4]), using memory mechanisms ([5], [6]) and multi-population schemes ([7]). The global functioning of EAs designed to deal with dynamic applications had inherited most of the characteristics of EAs used for static domains: the adjustment of mutation rate, the use of typical values for the crossover operator and the specification of unchanging population sizes. Even, when an explicit memory is used, its size is also set at the beginning and is usually a small percentage of the main population size. Less attention has been devoted to study the influence that population size can have in the performance of the EA.

Some research has been made about this issue: Schönemann ([8]) studied the impact of population size in the context of Evolutionary Strategies for dynamic environments and the results showed that the choice of the population size can affect the algorithm's performance. Simões et al. ([9]) proposed an EA to deal with changing environments which controls the size of population and memory during the run. Results from that work show that the performance of the EA can be considerably improved. Recently, Richter et al. ([10]) proposed a memory-based abstraction method using a grid to memorize useful information and the results obtained suggest that an optimal grid size may depend on the type of dynamics.

In this work we describe an extensive empirical study whose focus was the performance of distinct memory-based EAs that faced different environmental characteristics. The study is based on two benchmark problems and is divided in two parts: in the first part, we kept the populations sizes fixed and ran the EAs for different values for these parameters in order to analyze the impact of the chosen values on the EA's performance. In the second part, an EA using population and memory whose size may vary is ran and compared with the results of the previous experimentation. We show that the values set to population and memory sizes can have a great influence in the EA's performance. The use of an algorithm capable of dynamically adjusting the population and memory sizes during the run outperforms, in general, all the scenarios analyzed using constant populations sizes. These results are statistically supported. Due to space restrictions, we will not be able to show the complete and detailed description of this work. For more details see [11]. The rest of the paper is organized as follows: next section briefly reviews memory schemes for EAs in dynamic environments. Section 3, describes the three implemented memory-based EAs and Section 4 details the dynamic test environments used for this study. The experimental results and the analysis are presented in section 5. Section 6 concludes the paper and some considerations are made about future work.

2 Memory-Based Evolutionary Algorithms for Dynamic Environments

EAs using memory schemes work by storing information about the different environments that may occur and selectively retrieving that information later when another change in the environment is detected. Information can be stored

in memory implicitly or explicitly ([12]). In the case of implicit memory the stored information is kept in redundant representations of the genotype. There are some variants, e.g., using diploid (or multiploid) representations ([13]) or dualism mechanisms ([14]). In explicit memory schemes an extra space, called **memory population**, is used. Its main goal is to store useful information about the current environment and possibly reuse it each time a change occurs. It may also permit the main population to move to a different area in the landscape in one step, which in principle is harder when we rely only on standard genetic operators. ([12]).

Considerations about Memory: When using memory-based EA some questions can be asked concerning memory: 1) when and which individuals to store in memory, 2) which size to choose, 3) which individuals should be replaced when memory is full, 4) which individuals should be selected from memory and introduced in the main population when a change happens ([12]). In explicit memory schemes, the information stored in memory corresponds to the current best individual of the main population. Other approaches, together with this information, also keep information about the environment. The size of the memory is an important issue. Since the size of the memory is limited, it is necessary to decide which individuals should be replaced when new ones need to be stored. This process is usually called *replacing strategy*. Several replacing schemes have been proposed ([5], [9], [6]). The retrieval of memory information when a change occurs depends on what was stored. In some cases, memory is re-evaluated and the best individuals of memory replace the worst of the main population. Other approaches use the stored environmental information together with the best memory individual to create new information to introduce in the population when a change happens.

Population and Memory Sizes: Usually, memory-based EAs for changing environments use a memory of small size, when compared with the dimension of the main population. In most cases, the dimension of memory is chosen between 5% and 20% of the population size, with 10% the most chosen one, as can be seen in the used values in several studies listed in [11]. Choosing a constant value for population and memory sizes is widely used in memory-based EAs. Memory is always seen as playing a secondary role in the process, and is used always with a smaller dimension.

3 Description of the Implemented Memory-Based EAs

We choose to implement a memory-immigrant based EA (MIGA) and a direct memory EA (MEGA). Both use population and memory with unchanging sizes. An additional algorithm, using a direct memory scheme with changing population and memory sizes was implemented (VMEA), ran and compared with the other two. The three algorithms will be briefly described.

Memory-Immigrants Genetic Algorithm: The Memory-Immigrants Genetic Algorithm (MIGA) uses a direct memory scheme. It was proposed by

([15]) and works in the following way: the algorithm evolves a population of individuals in the standard evolutionary way: selection, crossover and mutation. Additionally, a memory is initialized randomly and used to store the current best individual of the population, replacing one of the initially randomly generated individuals, if it exists, or replacing the most similar in memory if it is better. Every generation the best individual in memory is used to create a set of new individuals, called immigrants that are introduced into the population replacing the worst ones. These new individuals are created mutating the best solution in memory using a chosen mutation rate. When a change is detected it is expected that the diversity introduced in the population by adding these immigrants can help the EA to readapt to the new conditions.

Memory-Enhanced Genetic Algorithm: The Memory-Enhanced Genetic Algorithm (MEGA) ([15]) is an adaptation of Branke's algorithm ([5]) and evolves a population of n individuals through the application of selection, crossover and mutation. Additionally, a memory of size m is used, starting with randomly created individuals. From time to time memory is updated in the following way: if any of the initial random individuals still exist, the current best solution of the population replaces one of them arbitrarily; if not, the most similar memory updating strategy is used to choose which individual to exchange. Memory is evaluated every generation and, when a change is detected, the memory is merged with the best $n - m$ individuals of the current population to form the new population, while memory remains unchanged.

Variable-size Memory Evolutionary Algorithm: Simões and Costa ([9]) proposed a Variable-size Memory Evolutionary Algorithm (VMEA) to deal with dynamic environments. This algorithm uses a population that searches for the optimum and evolves as usual, through selection, crossover and mutation. A memory population is responsible for storing good individuals of the main population at several moments of the search process. The two populations - main and memory - have variable sizes that can change between two boundaries. The basic idea of VMEA is to use the limited resources (total number of individuals) in a flexible way. The size of the two populations can change but the their sum cannot go beyond a certain limit. If an environmental modification is detected, the best individual of the memory is introduced into the population. In the case of either the population size or the sum of the two populations has reached the allowed maximum, the best individual in memory replaces the worst one in the current population. The algorithm was compared with other memory-based schemes using the standard dimensions for population and memory and the results validate its effectiveness.

4 Experimental Design

4.1 Dynamic Test Environments

The dynamic environments to test our approach were created using Yang's Dynamic Optimization Problems (DOP) generator ([16]). This generator allows

constructing different dynamic environments from any binary-encoded stationary function using the bitwise exclusive-or (XOR) operator. The characteristics of the change are controlled by two parameters: the speed of the change, r , which is the number of generations between two changes, and the magnitude of the change, ρ that controls how different is the new environment from the previous one.

In this work we constructed 16 cyclic DOPs, setting the parameter r to 10, 50, 100 and 200. The ratio ρ was set to different values in order to test different levels of change: 0.1 (a light shifting) 0.2, 0.5 and 1.0 (severe change). This group of 16 DOPs was tested using the three different algorithms with two benchmark problems, using ten different combinations os population and memory sizes. A total of 960 different situations were tested in this work. The selected benchmark problems, used to test the different EAs with different parameter settings were the dynamic Knapsack problem and the Onemax problem. The **Knapsack problem** is a NP-complete combinatorial optimization problem often used as benchmark. It consists in selecting a number of items to a knapsack with limited capacity. Each item has a value (v_i) and a weight (w_i) and the objective is to choose the items that maximize the total value without exceeding the capacity of the bag (C). We used a Knapsack problem with 100 items using strongly correlated sets of randomly generated data. The **Onemax problem** aims to maximize the number of ones in a binary string. So, the fitness of an individual is equal to the number of ones present in the binary string. This problem has a unique solution. In our experiments we used individuals of length 300. For more details see [11].

4.2 Parameters Setting

The EA's parameters were set as follows: generational replacement with elitism of size one, tournament selection with tournament of size two, uniform crossover with probability $p_c = 0.7$ and flip mutation with probability $p_m = 0.01$. Binary representation was used with chromosomes of size 100 for the Knapsack and 300 for Onemax problem. The probability of mutation and the ratio of immigrants introduced in population used in MIGA were 0.01 and 0.1, respectively, as suggested by [15].

All simulations used a global number of individuals equal to 100. These individuals were divided in main population and memory, in the following way: the main population used a population size between 10 and 90 individuals, in increases of 10 (10, 20,..., 90). Memory size was calculated with the remaining individuals: $M(\text{size}) = 100 - P(\text{size})$. The *generational* replacing strategy proposed by [6] was used in all EAs. For each experiment of an algorithm, 30 runs were executed. Each algorithm was run for a number of generations corresponding to 200 environmental changes. The overall performance used to compare the algorithms was the best-of-generation fitness averaged over 30 independent runs, executed with the same random seeds.

5 Results

In this section we will show partial results concerning: *(a)* the overall performance of the studied algorithms; *(b)* the adaptability of the algorithms along time, *(c)* the diversity of population and memory for the different algorithms and *(d)* the variation of population and memory sizes in VMEA.

The global results regarding MIGA and VMEA in the Knapsack problem under different environments are shown in Figure 1. VMEA's scores are plotted in the upper right corner. As we see, depending on the change period and the change ratio, the choice of population and memory sizes affects the algorithm's performance. For the Knapsack problem, using MIGA, the best choice was population with 60 or 50 individuals (and memory of 40 and 50 respectively), for dynamics changing rapidly ($r = 10$). In environments with slower changes ($r = 50$, $r = 100$, $r = 200$), the best results were achieved with population size of 80 or 70 and memory with 20 or 30 individuals, respectively. Either smaller or larger populations lead to a decrease of the EAs' performance. For the Onemax problem, MIGA obtained best results using population size of 40 individuals using faster changes and lower change ratios ($\rho = 0.1$ and $\rho = 0.2$) and population with 60 individuals for larger change ratios ($\rho = 0.5$ and $\rho = 1.0$). Increasing the change ratio and the change period, larger populations are required to achieve the best results: 70 to 90 individuals. Observing the graphics, it is visible the influence that population and memory sizes can have in the performance of this algorithm. It is evident there must be a tradeoff between memory and population sizes. The usual split of 10% for the memory size and 90% for the population size is not the best choice. In all cases, VMEA outperformed all instances of MIGA and MEGA (not shown here for lack of space), demonstrating robustness and adaptability in the studied problems under different environmental characteristics. As happens in the stationary environments ([2]) it is possible that different sizes of population and memory might be optimal at different stages of the evolutionary process.

The statistical validation of the obtained results used paired one-tailed t-test at a 0.01 level of significance and can be found in [11]. Statistical results support that choosing different sizes for the population and memory affects the algorithm's performance and this difference is statistically significant. In general, VMEA performed significant better than MIGA's and MEGA's best results on most dynamic environments. These results validate our expectation of the impact that a bad choice of population and memory sizes can have in EA's performance and also that dynamically sizing approaches should be further investigated.

Figure 2 shows an example of how the different algorithms evolved through the entire run for the Knapsack problem. These results were obtained using $\rho = 0.5$. For the other cases, the results were analogous. In all cases the difference in the algorithm's performance using a "good choice" and a "bad choice" for the population and memory sizes is considerable, especially in rapidly changing environments. In MIGA's and MEGA's worst results, the evolution is slower and the best performance is only achieved when $r = 200$ and at the end of the process, since more time between changes is given to the algorithms. VMEA's performance is

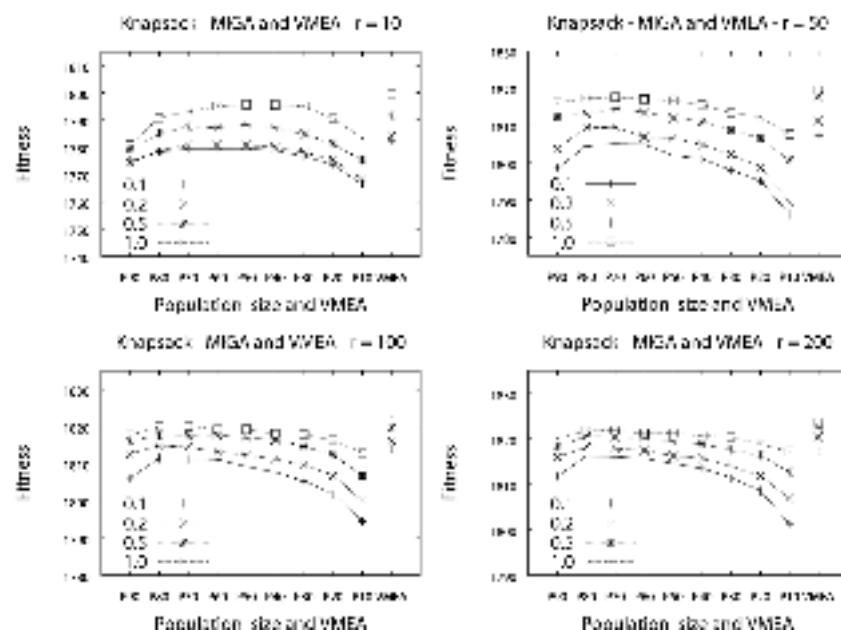


Fig. 1. Global results obtained in the dynamic **Knapsack** problem using MIGA (with different population and memory sizes) and VMEA

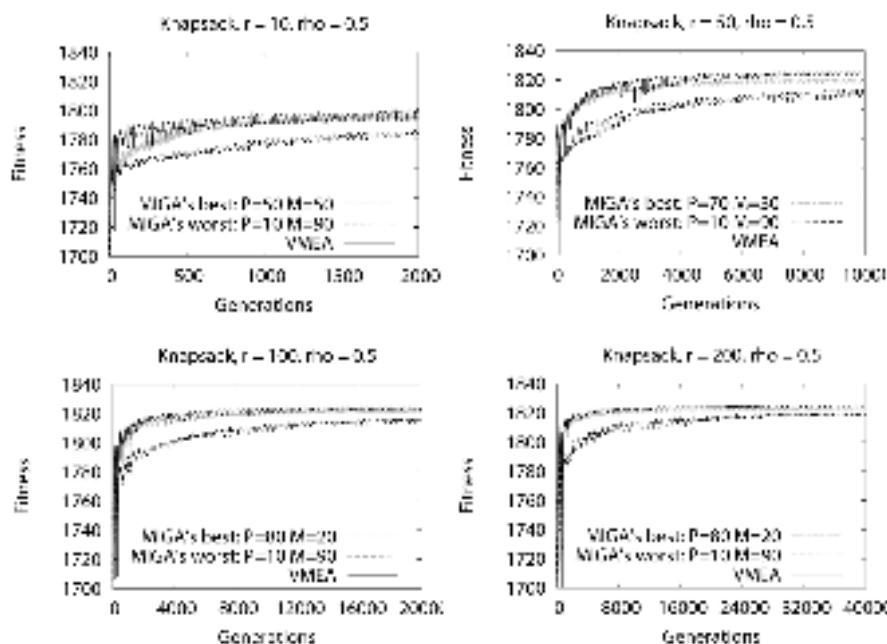


Fig. 2. Behavior of MIGA and VMEA in the **Knapsack problem**, $r = 10$, $r = 50$, $r = 100$ and $r = 200$, $\rho = 0.5$

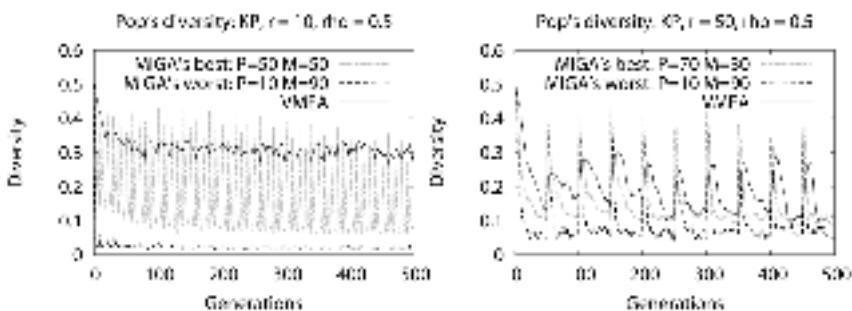


Fig. 3. Population's Diversity using MIGA and VMEA in the Knapsack problem, $r = 10$ and $r = 50$, $\rho = 0.5$

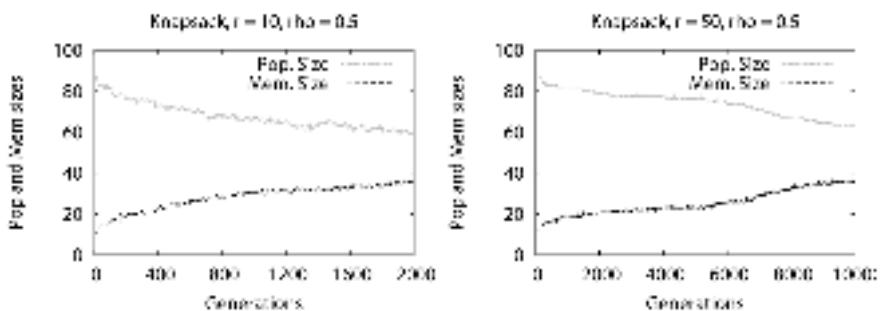


Fig. 4. Changes on Population and Memory Sizes using VMEA in Knapsack, $r = 10$ and $r = 50$, $\rho = 0.5$

always superior when compared with best performances of the other algorithms. To understand the previously shown results, we analyzed the diversity for different sizes of population and memory. Figure 3 shows the population's diversity using MIGA (best and worst results) and VMEA for the Knapsack problem. As we can see the worst performance of MIGA corresponds to the lower population's diversity and higher memory diversity. VMEA was the algorithm that maintains higher diversity in the population. In MIGA this observation was generalized to all situations. For MEGA the results are not clear and there are some cases where the best results were achieved by the population and memory sizes that maintained lower diversity. Thus, it's not straightforward that diversity is the reason for best or worst performances. Figure 4 shows the evolution of population and memory sizes during time using VMEA for different change periods in the Knapsack problem. The evolution of population and memory sizes has a typical behavior. In general memory tends to increase and population to decrease. As the number of generations increase this increase/decrease is slower. At the end of the process, population and memory sizes stabilize around similar values: population of 60-70 individuals and memory of 40-30 individuals, corresponding

to the set of values that often lead to best scores using MIGA and MEGA. The adjustment of population and memory sizes performed by VMEA can be considered "*blind*" since its only restrictions are the fixed boundaries and the global number of individuals. Nevertheless, superior results were attained.

6 Conclusions and Future Work

Usually, memory-based EAs dealing with dynamic environments use population and memory of constant sizes. Typically, memory size is set as a small percentage of population size. In this work we tried to understand if different population and memory sizes can have considerable influence in the performance of memory-based EAs dealing with different dynamic environments. Two direct memory EAs were ran with different values for population and memory sizes in two benchmark problems. A third algorithm, using dynamically adjusting population and memory sizes was also tested and compared with the other two algorithms. The obtained results show that the traditionally used values for population and memory sizes do not allow the best performance of the implemented EAs. It is clear that the tuning of the population and memory sizes has significant influence in the efficacy and convergence of the EAs. The best choice of values depends on the environmental characteristics, the problem to solve or the used algorithm. So, this choice is not linear or easy and trying to tune the population size before running the algorithm is practically impossible, since the combinations are huge and the process is time consuming. A preferred solution is the use of an EA capable of controlling the populations sizes during the run. In this work we used a simple EA which dynamically adjusts the population and memory dimensions and the results demonstrate its effectiveness and robustness to all environments and problems tested.

As future work we plan to investigate other sizing mechanisms in EAs to cope with dynamic environments. These approaches should take into account other information besides the limited number of the individuals that can exist, like, for example, the average fitness of population or aging mechanisms.

Acknowledgments

The work of the first author described in this paper was partially financed by the PhD Grant BD/39293/2006 of the Foundation for Science and Technology of the Portuguese Ministry of Science and Technology and High Education.

References

1. Gould, J.L., Keeton, W.T.: *Biological Science*. W. W. Norton & Company (1996)
2. Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*. Springer, Heidelberg (2003)

3. Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report TR AIC-90-001, Naval Research Laboratory (1990)
4. Yang, S.: Genetic algorithms with elitism-based immigrants for changing optimization problems. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 627–636. Springer, Heidelberg (2007)
5. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1875–1882. IEEE Press, Los Alamitos (1999)
6. Simões, A., Costa, E.: Improving memory's usage in evolutionary algorithms for changing environments. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), pp. 276–283. IEEE Press, Los Alamitos (2007)
7. Branke, J., Kaußler, T., Schmidt, C.: A multi-population approach to dynamic optimization problems. In: Parmee, I. (ed.) *Proceedings of Adaptive Computing in Design and Manufacture (ACDM 2000)*, pp. 299–308. Springer, Heidelberg (2000)
8. Schönenmann, L.: The impact of population sizes and diversity on the adaptability of evolution strategies in dynamic environments. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2004), vol. 2, pp. 1270–1277. IEEE Press, Los Alamitos (2004)
9. Simões, A., Costa, E.: Variable-size memory evolutionary algorithm to deal with dynamic environments. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 617–626. Springer, Heidelberg (2007)
10. Richter, H., Yang, S.: Memory based on abstraction for dynamic fitness functions. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008*. LNCS, vol. 4974, pp. 596–605. Springer, Heidelberg (2008)
11. Simões, A., Costa, E.: The influence of population and memory sizes on the evolutionary algorithm's performance for dynamic environments. Technical Report TR 2008/02, CISUC (2008)
12. Branke, J.: *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Dordrecht (2002)
13. Uyar, A.S., Harmancı, A.E.: A new population based adaptive dominance change mechanism for diploid genetic algorithms in dynamic environments. *Soft Computing* 9(11), 803–814 (2005)
14. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing* 9(11), 815–834 (2005)
15. Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Beyer, H.-G. (ed.) *Proceedings of the Seventh International Genetic and Evolutionary Computation Conference (GECCO 2005)*, vol. 2, pp. 1115–1122. ACM Press, New York (2005)
16. Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2006*. LNCS, vol. 3907, pp. 788–799. Springer, Heidelberg (2006)

Differential Evolution with Noise Analyzer

Andrea Caponio^{1,2} and Ferrante Neri¹

¹ Department of Mathematical Information Technology, University of Jyväskylä,
P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland

{andrea.caponio,neferran}@jyu.fi

² Department of Electrotechnics and Electronics, Technical university of Bari, Italy
caponio@deemail.poliba.it

Abstract. This paper proposes a Differential Evolution based algorithm for numerical optimization in the presence of noise. The proposed algorithm, namely Noise Analysis Differential Evolution (NADE), employs a randomized scale factor in order to overcome the structural difficulties of a Differential Evolution in a noisy environment as well as a noise analysis component which determines the amount of samples required for characterizing the stochastic process and thus efficiently performing pairwise comparisons between parent and offspring solutions.

The NADE has been compared, for a benchmark set composed of various fitness landscapes under several levels of noise bandwidth, with a classical evolutionary algorithm for noisy optimization and two recently proposed metaheuristics. Numerical results show that the proposed NADE has a very good performance in detecting high quality solutions despite the presence of noise. The NADE seems, in most cases, very fast and reliable in detecting promising search directions and continuing evolution towards the optimum.

1 Introduction

With numerical optimization in the presence of a noisy objective function, the most critical operation is, as highlighted in [1], selection of the most promising solutions for subsequent phases of the optimization algorithms. Obviously, the selection requires a prior fitness-based comparison operation whose reliability and significance can be heavily jeopardized by the noise. As summarized in [2], the noise in the fitness function causes two types of undesirable selection behavior: 1) a candidate solution may be underestimated and thus eliminated, 2) a candidate solution may be overestimated and thus saved and allowed to successively reproduce.

Although Evolutionary Algorithms (EAs) are in principle promising for optimization with noisy fitness functions, see [3], the noise still represents a challenge that needs to be handled, especially when the selection operations are performed.

Beginning with the earliest studies on this topic, it has been clear that two operations can be implemented in order to filter the noise and thus successfully perform pairwise comparisons between fitness values. The first operation consists of re-sampling (computing the fitness value) several times and then calculating

the average. The second operation consists of enlarging population size with the aim of reducing the risk of under and overestimation, see [4]. The topic, whether a re-sampling or an enlargement of the population size is more efficient, has been discussed in literature and various results supporting both philosophies have been presented, e.g., in [5] and [6].

In both cases, these actions lead to an increase in the amount of fitness evaluations with respect to a standard EA. This fact obviously leads to an increase in computational overhead, which can be unacceptable in real-world applications where the computational cost of each fitness evaluation may be high. Thus, in order to obtain efficient noise filtering without excessive computational cost, various solutions have been proposed in literature. In [7], two variants of adaptive re-sampling systems based on the progress of the evolution have been proposed. In [8], a variable sample size performed by means of a probabilistic criterion based on the solution quality is presented. In [9] both sample and population size are adaptively adjusted on the basis of a diversity measure.

Other works, e.g., [10] and [11], propose taking the fitness estimates of neighboring individuals in order to predict the fitness value of some candidate solutions. Paper [12], by employing a similar philosophy, proposes construction of a local linear surrogate model (an approximate model of the true fitness function) which locally performs the noise filtering.

Other works make some assumptions regarding the noise in order to propose integration of a filtering component within the selection schemes so as to perform sorting of the solutions without performing a large amount of samples. In [13], in order to reliably sort the fitness values in an Evolution Strategy (ES), a threshold during deterministic selection is imposed. Papers [1] and [14] propose sequential sampling procedures which aim at performing a massive re-sampling only when it is strictly necessary (fitness values are close to each other).

In recent years, noise filtering components have been designed and integrated into metaheuristics as opposed to the classical ES and Genetic Algorithms (GAs). For example, in [15], a sequential sampling procedure has been proposed for a Particle Swarm Optimization (PSO). In [16] and [17] very accurate statistics-based procedures for noise handling are integrated within PSO structures.

Some noise handling components have also been proposed for Differential Evolution (DE). In several papers, e.g. in [18], it is empirically shown that the DE is not recommended for noisy problems. However, recent works propose some modifications of the DE schemes which make it very competitive with advanced metaheuristics tailored to optimization in a noisy environment. According to the explanation given in [19] and [20], DE is based on an overly deterministic structure for handling difficulties imposed by the noise. Thus, introduction of stochastic elements in the framework (the scale factor, in this case) can greatly improve the DE performance in the presence of uncertainties. By following a similar logic, in [21], an opposition based DE (i.e., a DE which performs extra sampling of symmetrical points) is proposed for noisy environment and shows that generation of the opposition based points beneficially perturbs determinism of a DE structure in the presence of a noisy fitness.

In addition, it must be observed that a DE employs, as a survivor selection scheme, the so called one-to-one spawning of the solutions, i.e., replacement occurs pairwise between the parent and offspring solution. This fact, according to our interpretation, is at the same time the weak and strong point of a DE in a noisy environment. More specifically, if the noise disturbs the pairwise comparison and leads to an incorrect evaluation of the most promising solution between parent and offspring, the entire process can be jeopardized since incorrect directions of the search are transmitted over future generations. On the other hand, a pairwise comparison is easier to be performed, in the presence of the noise, than a ranking (as in a GA or an ES) and is thus a proper component which rather reliably allows performance of this pairwise comparison, significantly improving the DE performance.

This paper proposes a novel DE scheme which combines an increase of the stochastic pressure and a noise analysis while performing the selection in order to obtain an optimization algorithm with reliable behavior and high performance. Section 2 gives a description of the proposed DE based algorithm. Section 3 shows the numerical results on a set of numerous and various test functions. Section 4 summarizes the findings of this work and gives the conclusion.

2 Noise Analysis Differential Evolution

As shown in [4], a noisy fitness function (affected by Gaussian noise) can mathematically be expressed as:

$$\tilde{f}(x) = \int_{-\infty}^{\infty} [f(x) + z] p(z) dz = f(x), z \sim N(0, \sigma^2) \quad (1)$$

where x is the design vector, $f(x)$ is a time-invariant function and z is an additive noise normally distributed with 0 mean and variance σ^2 .

In principle, search of the optimum consists of optimizing $f(x)$; however since only the fitness values related to the $\tilde{f}(x)$ are available, the noisy optimization problem consists of optimizing $\tilde{f}(x)$ in a decision space D where x is defined. Without a loss in generality, this paper will refer to minimization problems.

In order to perform the minimization of $\tilde{f}(x)$, a novel DE based algorithm is proposed in this paper. The proposed algorithm, namely Noise Analysis Differential Evolution (NADE) consists of the following steps.

An initial sampling of S_{pop} individuals is performed pseudo-randomly with a uniform distribution function within the decision space D . Two extra values are associated with each candidate solution. These values are the number of samples n_{s_i} carried out on the corresponding solution x_i (n_{s_i} is initially set equal to 1 for all the individuals) and the resulting average fitness \bar{f}_i .

At each generation, for each individual x_i of the S_{pop} , three individuals x_r , x_s and x_t are pseudo-randomly extracted from the population. According to DE logic, a provisional offspring x'_{off} is generated by mutation according to the so called DE/rand/1 scheme:

$$x'_{off} = x_t + F(x_r - x_s) \quad (2)$$

where the scale factor F is, following suggestions given in [19] and [20], a pseudo-randomly selected value between 0.5 and 1.

Then, each gene of the new individual x'_{off} is switched with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{i,j} & \text{if } rand(0,1) < CR \\ x'_{off,j} & \text{otherwise} \end{cases} \quad (3)$$

where $rand(0,1)$ is a random number between 0 and 1; j is the index of the gene under examination. The resulting offspring x_{off} is evaluated and, the value $\delta = |\bar{f}(x_i) - \bar{f}(x_{off})|$ is computed. If $\delta > 2\sigma$ the candidate solution displaying the best performance value is simply chosen for the subsequent generation. This choice can be justified considering that for a given Gaussian distribution, 95.4% of the samples fall in an interval whose width is 4σ and has in its center the mean value of the distribution, see [22]. In this case, if the difference between two fitness values is greater than 2σ , likely that point which seems to have a better fitness is truly the best performing of the two candidate solutions.

On the other hand, if $\delta < 2\sigma$, the noise bands related to the two candidate solutions do overlap, and determining a ranking based on only one fitness evaluation is impossible. In this case, indicating with $\alpha = \min \{\bar{f}(x_i), \bar{f}(x_{off})\}$ and $\beta = \max \{\bar{f}(x_i), \bar{f}(x_{off})\}$, the following index is calculated:

$$v = \frac{\alpha + 2\sigma - (\beta - 2\sigma)}{\beta + 2\sigma - (\alpha - 2\sigma)}. \quad (4)$$

The index v represents the intersection of two intervals, characterized by a center in the fitness value and semi-width 2σ , with respect to their union. In other words, v is a normalized measure of the noise band overlap. This index can vary between 0 and 1. The limit condition $v \approx 0$ means that the overlap is limited and thus the pairwise ranking given by the single sample estimations is most likely correct. The complementary limit condition, $v \approx 1$ means that the interval overlap is almost total and the two fitness values are too close to be distinguished in the noisy environment. In other words, v can be seen as a reliability measure of a pairwise solution ranking in the presence of noisy fitness.

For the sake of clarity, a graphical representation of v is given in Fig. 1.

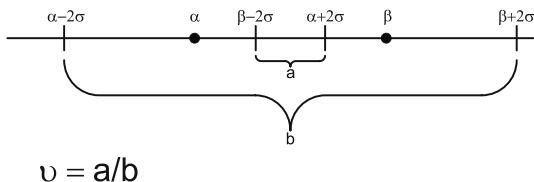


Fig. 1. Graphical representation of the parameter v

On the basis of the calculated value of v a set of additional samples n_s is performed for both parent and offspring solutions and their respective values of n_{s_i} and \bar{f}_i updated. These samples are determined by calculating:

$$n_s = \left\lceil \left(\frac{1.96}{2 \cdot (1 - v)} \right)^2 \right\rceil \quad (5)$$

where 1.96 is the upper critical value of a normal distribution associated to a confidence level equal to 0.95, see [22]. Thus, n_s represents the minimum amount of samples which ensure a reliable characterization of the noise distribution, i.e., the amount of samples which allows consideration of the average fitness values as the mean value of a distribution.

More specifically, each parent and offspring solutions are associated to a respective interval whose center is their fitness value and semi-width 2σ . According to the idea proposed in this paper, the amount of samples must be linearly increased with the overlapping of these two intervals. However, as shown in eq. (5), since for $v \rightarrow 1$ it would result in $n_s \rightarrow \infty$, a saturation value for n_{s_i} (i.e., the maximum amount of samples for each solution) has been set in order to avoid infinite loops. It must be remarked that this saturation value is the only extra parameter to be set, with respect to a standard DE. In addition, setting of this parameter can be intuitively carried out on the basis of the global computational budget available and the precision requirement in the specific application.

When additional samples are performed, the average fitness values are recalculated and the solution characterized by the most promising average fitness is selected for subsequent generation.

3 Numerical Results

The NADE has been tested on a set of various test functions and compared with the Differential Evolution with Randomized Scale Factor and Threshold based Selection (DE-RSF-TS) proposed in [20], the Durations Sizing Genetic Algorithm (DSGA) proposed in [7] and the Particle Swarm Optimization (PSOOHT) proposed in [16].

The DE-RSF-TS has been run with F pseudo-randomly selected between 0.5 and 1, and $CR = 0.3$. The threshold value τ has been set equal to 2σ . The DSGA has been set, following suggestions in [7], to always retain the best individual, to use a stochastic uniform selection, a single point crossover with probability 0.7, a gaussian mutation with probability 0.3, a $\gamma = 0.005$ and a $n_0 = 1$. The PSOOHT has been run with a maximum velocity equal to 20% of the search space dimension along each axis and an initial number of evaluations for each particle equal to 5. Furthermore, in order to avoid never ending evaluations in PSOOHT, the algorithm has been set to avoid evaluating each individual more than 30 times and to avoid performing more than $10 \times S_{pop}$ evaluations in a single iteration. The NADE has been run with F pseudo-randomly selected between 0.5 and 1, and $CR = 0.3$. The maximum number of evaluations per each individual

Table 1. Test Problems

Test Problem	n	Test Problem	Decision Space
Ackley	30	$-20 + e + \exp\left(-\frac{0.2}{n} \sqrt{\sum_{i=1}^n x_i^2}\right)$ $-\exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi \cdot x_i) x_i\right)$	$[-1, 1]^n$
Camelback	2	$4x_1^2 - 2.1x_1^2 + \frac{x_1^6}{3} + x_1 x_2 - 4x_2^2 + 4x_2^4$	$[-3, 3], [-2.2]$
Michalewicz	30	$-\sum_{i=1}^n \sin x_i \left(\sin \left(\frac{i \cdot x_i^2}{\pi} \right) \right)$	$[0, \pi]^n$
Rastrigin	30	$10n + \sum_{i=0}^n (x_i^2 - 10 \cos(2\pi x_i))$	$[-5.12, 5.12]^n$
Rosenbrock	30	$\sum_{i=1}^{n-1} \left((x_{n+1} - x_i^2)^2 + (1 - x_i)^2 \right)$	$[-2.048, 2.048]^n$
Schwefel	30	$\sum_{i=1}^n x_i \sin \left(\sqrt{ x_i } \right)$	$[-500, 500]^n$
Sum of powers	30	$\sum_{i=1}^n x_i ^{i+1}$	$[-1, 1]^n$

Table 2. Average fitness \pm standard deviation

Test Problem	DE-RSF-TS	DSGA	PSOOHT	NADE
Ackley	3.7542 ± 0.6552	17.130 ± 1.2312	18.986 ± 1.0613	0.4919 ± 0.1478
	20.833 ± 0.7138	19.800 ± 1.7008	19.763 ± 1.2447	13.110 ± 8.4052
	21.334 ± 0.2938	21.201 ± 0.1966	21.080 ± 0.2164	21.436 ± 0.1093
Rot. Ackley	3.9557 ± 0.8974	17.606 ± 1.0857	19.389 ± 0.9273	0.5282 ± 0.1501
	20.874 ± 0.8420	19.334 ± 1.7862	19.040 ± 1.9890	9.1067 ± 7.4108
	21.313 ± 0.2080	21.132 ± 0.2024	20.290 ± 1.2422	21.473 ± 0.2404
Camelback	-0.9566 ± 0.0656	-0.8849 ± 0.1397	-0.9887 ± 0.0403	-0.9894 ± 0.0338
	-0.6564 ± 0.3594	-0.7985 ± 0.2644	-1.0007 ± 0.241	-0.8946 ± 0.1531
	-0.3898 ± 0.4428	-0.4873 ± 0.6656	-0.8791 ± 0.1629	-0.8725 ± 0.1410
Michalewicz	-24.925 ± 0.6731	-13.469 ± 1.0047	-6.5073 ± 0.6902	-26.562 ± 0.7143
	-12.013 ± 3.5984	-13.209 ± 1.1053	-6.4515 ± 0.6655	-25.560 ± 0.8881
	-4.1756 ± 1.3652	-11.334 ± 2.0985	-5.6493 ± 0.9022	-16.743 ± 2.4328
Rot. Mich.	-5.3405 ± 0.5865	-4.9735 ± 0.8423	-4.0255 ± 0.2651	-5.6785 ± 0.8587
	-3.1619 ± 1.2090	-3.2201 ± 1.7405	-3.5428 ± 0.8367	-4.5397 ± 1.2298
	-1.7554 ± 1.2608	-1.3994 ± 1.2636	-3.1903 ± 0.9936	-2.6864 ± 1.5154
Rastrigin	63.113 ± 10.065	190.58 ± 24.652	202.40 ± 77.491	28.659 ± 6.5862
	126.37 ± 25.097	196.39 ± 29.074	57.979 ± 25.419	51.698 ± 13.394
	284.99 ± 54.065	208.78 ± 32.129	174.11 ± 72.302	92.150 ± 16.757
Rot. Rastr.	194.03 ± 15.8234	252.94 ± 37.067	349.29 ± 46.102	183.75 ± 15.309
	247.65 ± 23.049	265.41 ± 30.083	263.37 ± 34.203	213.07 ± 25.149
	319.15 ± 38.293	281.28 ± 45.454	282.85 ± 16.045	239.26 ± 33.649
Rosenbrock	$7.06e03 \pm 3.59e03$	$1.62e04 \pm 5.92e03$	$1.28e04 \pm 4.80e04$	$3.22e03 \pm 2.35e03$
	$1.3681e04 \pm 6.00e03$	$1.95e04 \pm 6.67e03$	$1.11e04 \pm 2.52e04$	$9.09e03 \pm 4.79e03$
	$2.99e04 \pm 1.48e04$	$3.21e04 \pm 8.38e03$	$3.65e03 \pm 2.21e04$	$1.81e04 \pm 1.16e04$
Schwefel	$2.60e03 \pm 454.57$	$5.34e03 \pm 544.53$	$6.01e03 \pm 515.22$	$2.13e03 \pm 461.89$
	$4.8694e03 \pm 703.23$	$5.43e03 \pm 502.62$	$6.73e03 \pm 792.84$	$2.52e03 \pm 516.79$
	$9.28e03 \pm 875.08$	$5.67e03 \pm 482.05$	$6.95e03 \pm 575.38$	$4.59e03 \pm 758.39$
Sum Pow.	0.0199 ± 0.0158	0.0157 ± 0.0085	0.5844 ± 0.5528	0.0065 ± 0.0057
	0.0398 ± 0.0261	0.0564 ± 0.0451	0.1973 ± 0.1568	0.0204 ± 0.0218
	0.0602 ± 0.0463	0.1235 ± 0.0854	0.1102 ± 0.1039	0.0374 ± 0.0300

Table 3. Results of the Student's t-test

		DE-RSF-TS	DSGA	PSOOHT
Ackley	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	=	-	-
Rotated Ackley	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	-	-	-
Camelback	$\sigma = 0.04$	+	+	=
	$\sigma = 0.10$	+	=	-
	$\sigma = 0.20$	+	+	=
Michalewicz	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	+	+	+
Rotated Michalewicz	$\sigma = 0.04$	=	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	+	+	=
Rastrigin	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	=
	$\sigma = 0.20$	+	+	+
Rotated Rastrigin	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	+	+	+
Rosenbrock	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	+	=	+
Schwefel	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	+	+	+
Sum of Power	$\sigma = 0.04$	+	+	+
	$\sigma = 0.10$	+	+	+
	$\sigma = 0.20$	+	+	+

has been set equal to 30. All the algorithms have been run with a population size of 30 individuals. The algorithms have been tested for the fitness functions listed in Table 1. It should be remarked that some rotated problems have been added to the benchmark set and thus a total amount of 10 fitness landscapes have been considered. The rotated problems are obtained by means of multiplication of the vector of variables to a randomly generated orthogonal rotation matrix.

For each test problem, codomain range C has been estimated as the difference between the fitness value in the global optimum (when analytically known, otherwise the best fitness value detected in literature) and the average fitness value computed over 100 points pseudo-randomly generated within the decision space. Then, for each test problem, three noisy test cases have been generated by adding to the time-invariant function a zero-mean Gaussian noise characterized by a standard deviation equal to 4%, 10% and 20% of C , respectively. The noise levels are indicated in Table 2 as $\sigma = 0.04$, $\sigma = 0.1$, and $\sigma = 0.2$, respectively.

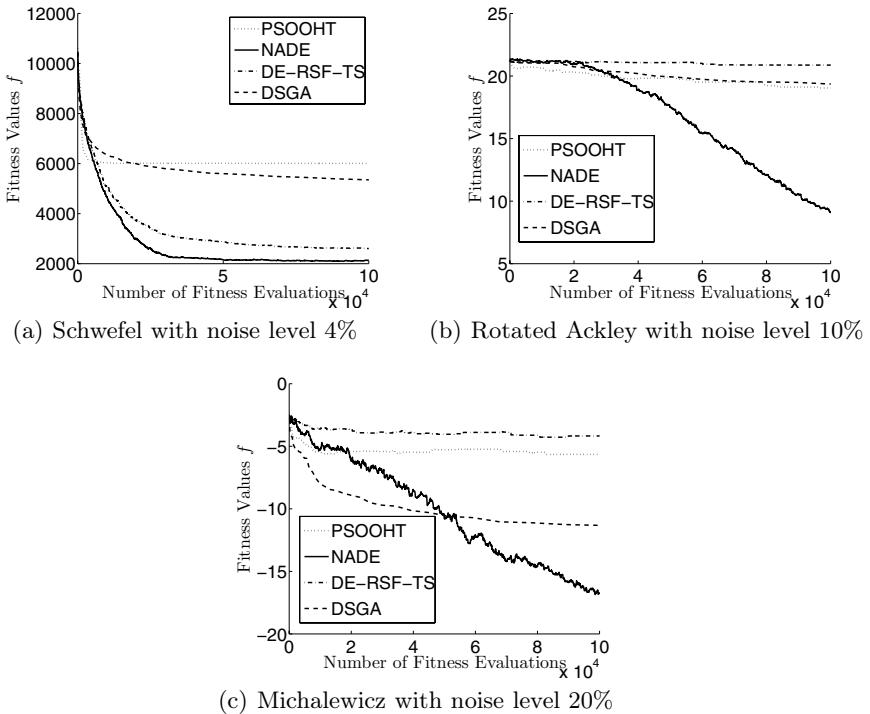


Fig. 2. Algorithmic performance over selected test problems

For the benchmark set composed of 30 optimization problems (10 test problems \times 3 noise levels), each algorithm has been run 30 times, for 100000 fitness evaluations. Thus, the same computational budget is assigned to each algorithm. While all optimization tests have been carried out in the presence of noise, actual progress in the optimization of true fitness function (f with reference to eq. (1)) has been saved in order to highlight actual performance of the algorithms in minimizing a landscape despite the noise. Table 2 shows the average (over the 30 independent runs) of the true fitness values, after 100000 evaluations, \pm the corresponding standard deviation values. The best results are highlighted in bold face.

In order to prove statistical significance of these results, the Student's t-test has been applied according to the description given in [22] for a confidence level of 0.95. Final values obtained by the NADE have been compared to the final value returned by each algorithm used as a benchmark. Table 3 shows results of the test. Indicated with "+" is the case when the NADE statistically outperforms, for the corresponding test situation, the algorithm mentioned in the column; indicated with "=" is the case when pairwise comparison leads to success of the t-test i.e., the two algorithms have the same performance; indicated with "-" is the case when the NADE is outperformed.

Numerical results in Table 2 and Table 3 show that the NADE has a very good performance in terms of final solutions. More specifically the NADE is outperformed in only six cases out of the ninety pairwise comparisons considered, the NADE has the same performance as the other algorithms in seven cases and outperforms the other algorithms in the remaining seventy-seven cases. Thus the NADE is outperformed in only 6.6% of the cases and outperforms the other algorithm in over the 85% of the cases. Figures 2 show the average performance trend (in terms of true fitness) on selected test problems.

4 Conclusion

A novel DE based algorithm for handling noisy fitness problems has been proposed. The proposed algorithm combines a randomization in the scale factor with a noise analysis in the selection scheme. The randomization is supposed to overcome the high determinism of DE, which negatively affects algorithmic behavior in the presence of noise. The noise analysis is supposed to accurately perform pairwise comparison with minimal computational expense. Numerical results show that the proposed NADE displays a very good performance on most of the test problems taken into account. More specifically, in the presence of noise equal to 4% of the co-domain width the proposed algorithm has an extremely good performance when compared with both classical noise filtering algorithms and modern metaheuristics for optimization in noisy environment. This result can be of interest for engineers and practitioners since this is the typical noise bandwidth for an industrial measurement instrument. Performance of the NADE in a highly noisy environment is also very promising and, for some fitness landscapes, significantly better than that obtained by the other algorithms under consideration.

References

1. Branke, J., Schmidt, C.: Selection in the presence of noise. In: Cantu-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 766–777. Springer, Heidelberg (2003)
2. Di Pietro, A., While, L., Barone, L.: Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1254–1261 (2004)
3. Beyer, H.G., Sendhoff, B.: Functions with noise-induced multimodality: a test for evolutionary robust optimization-properties and performance analysis. *IEEE Transactions on Evolutionary Computation* 10(5), 507–526 (2006)
4. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey
5. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic algorithms in noisy environments. *Machine Learning* 3, 101–120 (1988)
6. Beyer, H.G.: Toward a theory of evolution strategies: Some asymptotical results from the $(1, + \lambda)$ -theory. *Evolutionary Computation* 1(2), 165–188 (1993)
7. Aizawa, A.N., Wah, B.W.: Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation* 2(2), 97–122 (1994)

8. Stagge, P.: Averaging efficiently in the presence of noise. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 188–200. Springer, Heidelberg (1998)
9. Neri, F., Casella, G.L., Salvatore, N., Kononova, A.V., Acciani, G.: Prudent-daring vs tolerant survivor selection schemes in control design of electric drives. In: Rothlauf, F., et al. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 805–809. Springer, Heidelberg (2006)
10. Branke, J., Schmidt, C., Schmeck, H.: Efficient fitness estimation in noisy environments. In: Spector, L., et al. (eds.) Genetic and Evolutionary Computation Conference, pp. 243–250. Morgan Kaufmann, San Francisco (2001)
11. Sano, Y., Kita, H., Kamihira, I., Yamaguchi, M.: Online optimization of an engine controller by means of a genetic algorithm using history of search. In: Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning, pp. 2929–2934. Springer, Heidelberg (2000)
12. Neri, F., del Toro Garcia, X., Casella, G.L., Salvatore, N.: Surrogate assisted local search on PMSM drive design. *COMPEL: International Journal for Computation and Mathematics in Electrical and Electronic Engineering* 27(3), 573–592 (2008)
13. Markon, O., Arnold, D.V., Biick, T., Beielstein, T., Beyer, H.G.: Thresholding - a selection operator for noisy ES. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 465–472 (2001)
14. Branke, J., Schmidt, C.: Sequential sampling in noisy environments. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiño, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 202–211. Springer, Heidelberg (2004)
15. Bartz-Beielstein, T., Blum, D., Branke, J.: Particle swarm optimization and sequential sampling in noisy environments. In: Doerner, K.F., et al. (eds.) Metaheuristics. Operations Research/Computer Science Interfaces Series, vol. 39, pp. 261–273. Springer, Heidelberg (2007)
16. Pan, H., Wanga, L., Liu: Particle swarm optimization for function optimization in noisy environment. *Applied Mathematics and Computation* 181, 908–919 (2006)
17. Klamargias, A.D., Parsopoulos, K.E., Alevizos, P.D., Vrahatis, M.N.: Particle filtering with particle swarm optimization in systems with multiplicative noise. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 57–62. ACM, New York (2008)
18. Krink, T., Filipič, B., Fogel, G.B.: Noisy optimization problems - a particular challenge for differential evolution? In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 332–339 (2004)
19. Das, S., Konar, A.: An improved differential evolution scheme for noisy optimization problems. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 417–421. Springer, Heidelberg (2005)
20. Das, S., Konar, A., Chakraborty, U.: Improved differential evolution algorithms for handling noisy optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1691–1698 (2005)
21. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution for optimization of noisy problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1865–1872 (2006)
22. NIST/SEMATECH: e-handbook of statistical methods,
<http://www.itl.nist.gov/div898/handbook/>

An Immune System Based Genetic Algorithm Using Permutation-Based Dualism for Dynamic Traveling Salesman Problems

Lili Liu¹, Dingwei Wang¹, and Shengxiang Yang²

¹ School of Information Science and Engineering, Northeastern University
Shenyang 110004, P. R. China

liulili1202@gmail.com, dwwang@mail.neu.edu.cn

² Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
s.yang@mcs.le.ac.uk

Abstract. In recent years, optimization in dynamic environments has attracted a growing interest from the genetic algorithm community due to the importance and practicability in real world applications. This paper proposes a new genetic algorithm, based on the inspiration from biological immune systems, to address dynamic traveling salesman problems. Within the proposed algorithm, a permutation-based dualism is introduced in the course of clone process to promote the population diversity. In addition, a memory-based vaccination scheme is presented to further improve its tracking ability in dynamic environments. The experimental results show that the proposed diversification and memory enhancement methods can greatly improve the adaptability of genetic algorithms for dynamic traveling salesman problems.

1 Introduction

Many real-world optimization problems are time-varying, i.e., the fitness function, design variables and the environmental conditions may change over time due to the arrivals of new tasks, or circumstance fluctuations, and so on. It is interesting to investigate dynamic instances when changes are not so severe that the new optimal solutions might be related to the old ones. As one of the most typical NP-hard combinatorial optimization problems, the traveling salesman problem in dynamic environments (DTSP), which is caused by deletion of some cities, the change of traffic costs between cities, or the addition of cities over time, has a practical application in production and traffic systems [6,7,8] and requires optimization algorithms to be able to track the optima in new situations.

Genetic algorithms (GAs) have been applied to address dynamic optimization problems (DOPs) in recent years with several promising results [12,15,21]. Intrinsically, GAs are motivated by the natural evolution, which is always subject to a changing environment. However, classic GAs should be modified to deal with the convergence problem for solving DOPs. The tendency of the standard

algorithm making all individuals eventually converge onto the promising individual of the current population may reduce the population diversity [1,2], which leads to a low exploration ability in dynamic environments. Therefore, several approaches have been developed into GAs to address DOPs, including random or guided immigrants [4,19] and hypermutation schemes [2].

Diversity and memory are two chief factors for enhancing the adaptability of organisms in the ever-changing environment in nature [20]. For example, the immune system of each animal body can generate antibodies to prevent invasions of detrimental pathogens [9]. As a major type of lymphocytes in the adaptive system, B-cells can be activated and cloned. The cloning process can be regarded as a special form of mutation, namely *somatic hypermutation*. In this way, the system holds various antibodies to neutralize antigens. Meanwhile, some B-cells can store the current information as memory in order to immediately respond to the same or similar pathogens. Hence, the diverse antibodies and the stored information help the immune system to detect and protect organisms against potentially harmful pathogens in different situations.

Following the above discussions, GAs with effective diversification and memory strategies inspired by the immune system, which can improve the exploration capacity and fully utilize the memorized information, may have a good potential in dynamic environments. Simões and Costa [18] proposed an immune system based GA, called *ISGA* for dynamic environments in the binary search space. It works by performing a transformation operation, which can be considered as the somatic hypermutation of B-cells, together with the memory scheme to allow the population rapidly respond to the changing landscapes. Yang [20] further investigated a variant of ISGA that adopts some new techniques, which greatly improve ISGA's adaptivity in tracking the changing optima.

This paper extends the idea of the immune system based GAs to solve DTSPs. A permutation-based dualism scheme, which has proven to be a useful diversification technique for DOPs in the natural number-encoded space [15], is introduced in the clone process of the immune system based GA for DTSPs. The resultant GA is denoted PISGA. Moreover, a memory-based vaccination strategy is proposed with the expectation of evolving individuals by taking advantage of information stored in the past evolutionary process, which may lead to a better tracking capacity in dynamic environments. The effect of some key techniques on the performance of PISGA for DTSPs are experimentally investigated. Furthermore, the performance comparison with other GA variants validates the efficiency of PISGA for DTSPs.

2 Relevant Work on the Dynamic TSP

Dynamic TSP was firstly introduced by Psaraftis in [16]. In recent years, some variants of classic intelligent optimization methods have been developed for a promising behavior on DTSPs. Guntsch and Middendorf [8] introduced a population-based ant colony optimization algorithm to solve DTSPs. They applied and compared three strategies for modifying the pheromone information

instead of the restart strategy in reaction to the change of problem instances. Eyckelhof and Snoek [3] created DTSP instances by introducing “traffic jam” via increasing edge weights on the current best paths, and suggested that it is crucial to utilize the previous information for enhancing the search ability of ant systems in dynamic environments.

As a class of stochastic optimization technique that mimics the selection and modification principles in natural evolution, evolutionary algorithms (EAs) have attracted an increasing concern to address DTSPs. When addressing DTSPs, an efficient EA optimizer should show continuous adaption to track the changing optimal routes. Huang [10] carried out a thorough comparison between DTSP and some ordering problems to illustrate the universality and availability of this problem in real-world applications. The experimental results suggest that EAs might be a good candidate for DTSPs. Zhou [22] presented three types of operators based on some prior knowledge of the current environment to enhance the search ability of EAs for DTSPs. Li et al [13] further improved the inver-over operator based on a gene pool, which stores a set of most promising gene segments by applying some heuristic rules. These approaches indicate again that adopting effective diversity maintaining or enhancing schemes and memory-based methods may provide EAs with a competitive performance to solve DTSPs.

3 The Proposed Approach: PISGA

In this paper, a permutation-based dualism scheme and some mechanisms of the immune system are integrated into the standard GA for DTSPs. The proposed PISGA has the following three characteristics:

1. It shares the basic framework of the ISGA proposed in [20];
2. The diversification scheme utilizes the permutation-based dualism to simulate the somatic hypermutation when performing the clone operation;
3. The memory-based scheme employs a vaccination method by extracting and conveying valuable information according to the stored knowledge.

The pseudo-code of PISGA is outlined in Fig. 1. In the following subsections, the key operators and the framework of PISGA are described in details.

3.1 The Permutation-Based Dual Operator

In nature, some organisms can utilize pieces of DNA or gene segments from their complements to fully integrate diverse functions in response to different environments. Based on the idea that one organism or individual may have several complements or dual entities based on different attributes, a permutation-based dualism was firstly proposed to GAs in [15]. The principle of the dualism scheme is illustrated in Fig. 2. Similar to diploid mechanisms in GA, through the dual mapping based on different attributes, each individual has several dual entities regarding different attributes. This may promote the population diversity to a reasonable level.

```

 $t := 0$ 
Initialize population  $P(0)$  and  $M(0)$  randomly
repeat
    Evaluate population  $P(t)$  and  $M(t)$ 
    Replace the worst individual in  $P(t)$  by the elite from  $P(t - 1)$ 
    if an environmental change is detected then
        Retrieve the best memory point  $B_M(t)$ 
        Perform vaccination operations for  $r_v * n$  worst individuals by Eq. (4)
         $t_M := t + \text{rand}(5, 10)$ 
    end if
    if  $t = t_M$  then
        Replace the worst individual in  $M(t)$  with the best member in  $P(t)$ 
    end if
     $P'(t) := \text{selectForReproduction}(P(t))$ 
    Performing clone operations for individuals in  $P'(t)$  using permutation-based dualism according to Eq. (3)
     $t := t + 1$ 
until the termination condition is met

```

Fig. 1. Pseudo-code of the PISGA, where t_M is the time point to update the memory

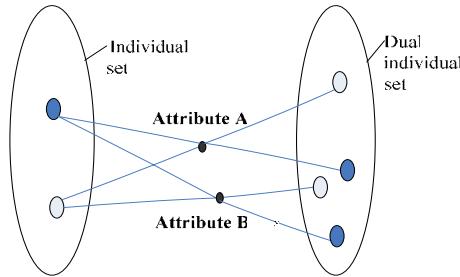


Fig. 2. Dual mapping based on different attributes

The concept of *permutation* is generally employed to investigate a particular family of groups, called *permutation group*, in the abstract algebra theory [5]. Since it can be dissolved into different cycles to represent collocation transformations, it is practical to introduce the permutation with some characteristics for decomposing the attributes. GAs with the permutation-based dual operator have been shown to outperform standard GA for solving some combinatorial optimization problems in dynamic environments [14,15].

Similar to the ISGA [20], PISGA starts from a population of B-cells. At each generation, individuals are selected to form the mating pool. Individuals in the mating pool will be randomly partitioned into two sub-populations, in which individuals are operated by the operator C_{dual} with respect to two attributes, namely discrete and inverse attribute respectively. They are described as follows:

1* *Discrete attribute*: when a m -bit chromosome S is transformed to S' by a permutation denoted as f_{d1} , not only all of their alleles are different, but also the neighborhood of each gene in S is shifted completely in S' . Permutation f_{d1} for discrete attribute is adopted as follows:

$$f_{d1} = \begin{cases} (1, 3, \dots, 2i-1, \dots, 2(m-i), \dots, 4, 2), & \text{if } m \text{ is even,} \\ (1, 3, \dots, 2i-1, \dots, 2(m-i+1), \dots, 4, 2), & \text{if } m \text{ is odd,} \end{cases} \quad (1)$$

where i denotes the i -th bit of the chromosome.

2* *Inverse attribute*: when a chromosome S is transformed to S' by a permutation denoted as f_{d2} , the order of genes in S is inverted in S' . The permutation f_{d2} for the inverse attribute is designed as follows:

$$f_{d2} = \omega'_1 \omega'_2 \dots \omega'_i \dots \omega'_{[m/2]} \quad (2)$$

where ω'_i is the transposition $(i, m - i + 1)$ and “[]” denotes rounding the inner value down to an integer.

Given the above definitions, a dual operator can be defined as follows:

$$C_{dual} = (\alpha_1, d'_1)(\alpha_2, d'_2) \dots (\alpha_s, d'_s) \quad (3)$$

where $\alpha_i \in [0, 1]$ is the dual factor and the transposition d'_i , called the *dual unit*, takes effect in the dual operation with a probability α_i . The selected individual is mapped onto a new individual by C_{dual} .

We adopt such an operator for two reasons. First, one permutation can be dissolved into several cycles. Each individual can be transformed by these cycles with the corresponding dual factors and a permutation can create k ($k \in [1, \sum_{i=1}^s C'_s]$) dual individuals. Hence, the diversity degree might be greatly improved even if the population has been convergent. Second, with the dual factor, some adjacency relationships of cities may be preserved into the next generation, which is expected to provide continuous tracking capacity over time.

3.2 The Memory-Based Vaccination

Jiao and Wang [11] introduced the vaccination which is abstracted from the prior knowledge of a problem, with the aim of accelerating the search speed for seeking the fixed optimum in a static context.

In this study, a novel vaccination extraction and injection technique based on the stored information is proposed in PISGA, which is also inspired by the permutation group theory. PISGA uses a memory of size N_m with a stochastic updating period (i.e., if an updating occurs at time t , the next time to update the memory is $t_M := t + \text{rand}(5, 10)$), and the vaccination operation is evoked when an environment change is detected. The detection method is similar to that used in Yang's ISGA [20]. That is, the stored individuals are re-evaluated for each generation, and a change is considered to occur if the fitness value of any individual does not match its memorized best value at the last generation. In PISGA, the stored individual which mostly matches the current environment

is retrieved and utilized to perform vaccination operations for the $r_v * n$ worst members, where n is the population size and r_v is the vaccination ratio. The vaccination operation is described below.

If we denote a chromosome as $S = (c_1, c_2, \dots, c_m)$ and the best memory solution as $S_m = (a_1, a_2, \dots, a_m)$, there exists a permutation f_v :

$$f_v : [c_1 | c_2 | \dots | c_m] \rightarrow [a_1 | a_2 | \dots | a_m]$$

where f_v is equal to a product of disjointed cycles. Hence, the vaccine with respect to the memory point S_m can be defined as follows:

$$C_{vaccine} = (\beta_1, v'_1)(\beta_2, v'_2) \dots (\beta_k, v'_k) \quad (4)$$

where $\beta_i \in [0, 1]$ is the vaccine-extraction probability and the cycle v'_i is the vaccine unit that takes effect in the vaccination operation with the probability β_i . The selected individual is transformed to a new individual by $C_{vaccine}$.

The reasons for introducing this scheme lies in two factors. First, it is difficult to obtain the knowledge of a problem in advance, especially for the real-world applications in dynamic contexts. The best memory point which may reflect the requirements of the current environment in some extent may be a good candidate for extracting vaccine for a better optima tracking capacity. Second, the injection operation considers part of “promising” cycles instead of all of them. It is expected to help individuals to take in valuable information and prevent converging to the best memory point.

4 Experimental Study

4.1 Experimental Design

In order to test the proposed PISGA, experiments were carried out on DTSPs to compare the performance of PISGA with other two variants of GAs for dynamic environments. The first GA, denoted *R-SGA*, re-initializes all individuals when a change is detected, which is a straightforward method to respond to dynamic environments. Another peer GA is the permutation-based dual GA (PBDGA) [15], which has proven to be competitive in some dynamic environments.

In this study, we adopted the method proposed in [8] to create DTSP instances based on the data of kroA100 [17], which is created by changing the costs between some cities. That is, randomly removing half of the cities from the standard TSP data set to create a *spare pool*. For every τ intervals, a set of c cities are randomly replaced by the locations in the spare pool. For the purpose of testing the adaption of algorithms under different dynamics, the environmental dynamics parameters were set to $c \in \{1, 10, 15\}$ and $\tau \in \{50, 100, 200\}$, where c determines the severity of changes and τ specifies the speed of changes. This gives 9 dynamic scenarios as listed in Table 1. For each experiment on one scenario, 10 environmental changes were allowed for each algorithm on a DTSP.

For all GAs, parameters were set as follows: the uniform cycle crossover with a probability $p_c = 0.6$, removal mutation with the mutation probability $p_m = 0.1$,

Table 1. The t -test results of comparing algorithms on dynamic problems

(τ, c)	(50,1)	(50,10)	(50,15)	(100,1)	(100,10)	(100,15)	(200,1)	(200,10)	(200,15)
Scenario	1	2	3	4	5	6	7	8	9
PISGA – R-SGA	+	+	+	+	+	+	+	+	+
PISGA – PBDGA	~	~	~	+	+	+	+	+	+

fitness proportionate selection with the roulette wheel and elitism strategies [10]. The population size is set to 100, including 10 individuals stored in the memory. For PISGA and PBDGA, all the dual probabilities and vaccination probabilities were set to 0.5 and the vaccination ratio r_v was set to 0.1.

The overall performance of algorithms is defined as follows [19]:

$$\overline{F_{BG}} = \frac{1}{NG} \sum_{i=1}^{NG} (F_{BGi}) \quad (5)$$

where $\overline{F_{BG}}$ is the mean best-of-generation fitness averaged across the whole evolutionary process, F_{BGi} is the best-of-generation fitness of generation i , NG is the total number of generations.

4.2 Experimental Results and Analysis

The experimental results for comparisons are provided in Fig. 3 on different dynamic instances. The corresponding statistical test results of comparing algorithms by one-tailed t -test with 98 degrees of freedom at a 0.05 level of significance are given in Table 1, where the result regarding Alg. 1 – Alg. 2 is shown as “+”, “-”, or “~” when Alg. 1 is significantly better than, significantly worse than, or statistically equivalent to Alg. 2 respectively.

From Fig. 3 and Table 1, it can be seen that PISGA outperforms R-SGA and PBDGA in most test cases. This result validates our expectation of the permutation-based dual mechanism and the memory-based vaccination approach in the immune-based GA for investigated DTSPs. Comparing PISGA and PBDGA, an interesting observation is that PISGA dominates on investigated DTSPs especially with low change frequencies (i.e., $\tau = 100$ and 200). The reason may lie in that sufficient search iterations which could produce a better memory point in the past is beneficial for tracking the optimum in a new environment. R-SGA performs the worst in all cases. This demonstrates that the combination of the dual mechanism and other approaches that enhance the exploitation ability in promising regions, such as the learning operator in PBDGA and the vaccination operator in PISGA, may lead to a better tracking behavior for GAs.

One technique introduced in PISGA is the permutation-based dual scheme. In order to investigate the effect of this operator on the performance of PISGA, experiments were implemented on PISGA with the dual probability α setting to 0, 1 and 0.5, which implies performing clone without dual operator, completely transforming the gene segments of individuals in the mating pool with respect to the two attributes, and partially transforming the gene segments with the probability of 0.5 respectively. Experimental results on the dynamic behavior of

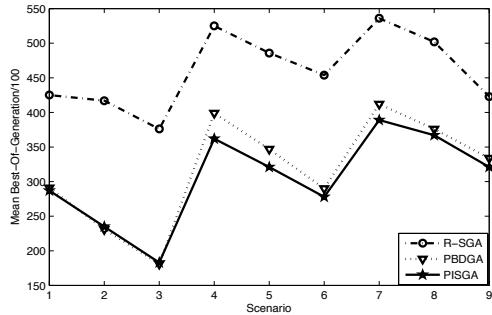


Fig. 3. Mean best-of-generation fitness for different DTSPs

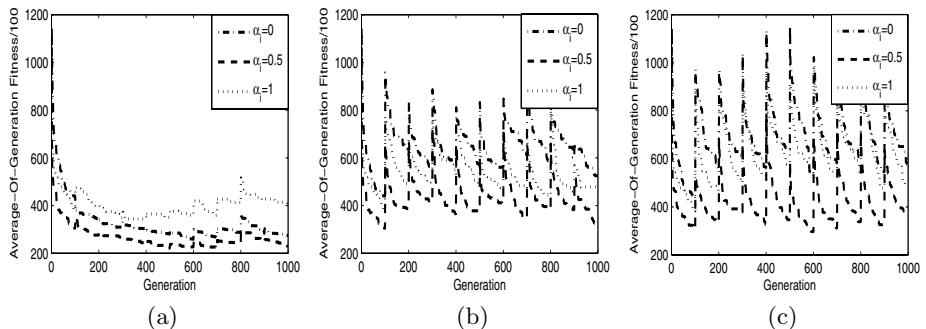


Fig. 4. Dynamic performance of PISGA with different α_i on investigated DTSPs with $\tau = 100$ and different severities of environmental changes: (a) $c = 1$, (b) $c = 10$, and (c) $c = 15$

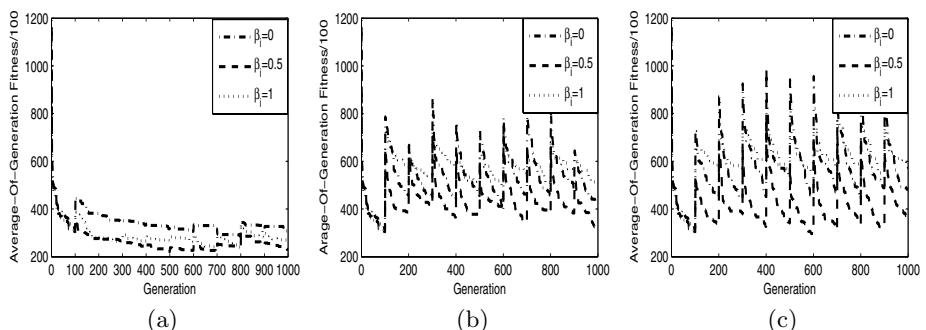


Fig. 5. Dynamic performance of PISGA with different β_i on investigated DTSPs with $\tau = 100$ and different severities of environmental changes: (a) $c = 1$, (b) $c = 10$, and (c) $c = 15$

PISGAs on investigated DTSPs with $\tau = 100$ are given in Fig. 4. It can be seen that the value of 0.5 always perform the best, indicating an advantage of the adopted dual mechanism over the other two settings for investigated DTSPs. PISGA with $\alpha = 0$ tends to strongly suffer from being trapped to local optima due to the diversity loss. The convergence curve of PISGA with $\alpha = 1$ shows that a reasonable diversity degree (i.e., not the larger, the better) is more helpful for a better performance in dynamic environments.

Furthermore, to examine the efficiency of the vaccination scheme on the behavior of PISGA, experiments were also carried out with the the value of β set to 0, 1, and 0.5 on the DTSPs. The results, presented in Fig. 5, show that the proposed memory-based approach is necessary and that a large β (i.e., $\beta = 1$) may weaken the search efficiency due to the diversity deficiency.

5 Conclusions and Future Work

This paper proposes an immune system based genetic algorithm to address dynamic traveling salesman problems. A permutation-based dualism scheme is introduced and integrated into the immune operation in order to enhance the exploration capacity of the population to watch over new optima. Furthermore, a memory-based vaccination strategy is presented to drive individuals to exploit promising regions based on the valuable information extracted from the memory. Experimental results show the efficiency of the proposed techniques for the immune-based GA for DTSPs.

There are several issues deserving the future work. A further study on the sensitivity of parameters on the performance of PISGA and a comprehensive comparison with other approaches for DTSPs are under investigation. It would be valuable to introduce other mechanisms in immune systems for GAs in dynamic environments. Another interesting topic is to extend our scope to more complex dynamic problems, such as dynamic vehicle routing problems.

Acknowledgments

The work by Lili Liu and Dingwei Wang was supported by the Key Program of National Natural Science Foundation (NNSF) of China under Grant No. 70431003 and Grant No. 70671020, the Science Fund for Creative Research Group of NNSF of China under Grant No. 60521003 and the National Science and Technology Support Plan of China under Grant No. 2006BAH02A09. The work by Shengxiang Yang was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant No. EP/E060722/1.

References

1. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proc. of the 1999 Congr. on Evol. Comput., vol. 3, pp. 1875–1882 (1999)
2. Cobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: Proc. of the 5th Int. Conf. on Genetic Algorithms, pp. 523–530 (1993)

3. Eyckelhof, C.J., Snoek, M.: In: ANTS 2002: Proc. of the 3rd Int. Workshop on Ant Algorithms, pp. 88–99 (2002)
4. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Proc. of the 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137–144 (1992)
5. Gilbert, W.J.: Modern Algebra with Application. John Wiley and Sons, Chichester (1976)
6. Guntsh, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: Proc. of the 2001 Genetic and Evol. Comput. Conf., pp. 860–867 (2000)
7. Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., et al. (eds.) EvoWorkshops 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
8. Guntsch, M., Middendorf, M.: A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoWorkshops 2002. LNCS, vol. 2279, pp. 72–81. Springer, Heidelberg (2002)
9. Hartl, D.L., Jones, E.W.: Genetics: Principles and Analysis. Jones and Bartlett Publishers, Inc. (1998)
10. Huang, Z., Hu, X., Chen, S.: Dynamic traveling salesman problem based on evolutionary computation. In: Proc. of the 2001 IEEE Congr. on Evol. Comput., pp. 1283–1288 (2001)
11. Jiao, L., Wang, L.: A novel genetic algorithm based on immunity. IEEE Trans. on Systems, Man, and Cybern. Part A: Systems and Humans 30(5), 552–561 (2000)
12. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - A survey. IEEE Trans. on Evol. Comput. 9(6), 303–317 (2005)
13. Li, C., Yang, M., Kang, L.: A new approach to solving dynamic traveling salesman problems. In: Wang, T.-D., Li, X.-D., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 236–243. Springer, Heidelberg (2006)
14. Liu, L., Wang, D., Wang, H.: A new dual scheme for genetic algorithm in dynamic environments. In: Proc. of the 2008 Control and Decision Conf., pp. 135–138 (2008)
15. Liu, L., Wang, D., Ip, W.H.: A permutation-based dual genetic algorithm for dynamic optimization problems. In: Soft Computing (published online on July 18, 2008)
16. Psaraftis, H.N.: Dynamic vehicle routing problems. In: Golden, B.L., Assad, A.A. (eds.) Vehicle Routing: Methods and Studies, pp. 223–248. Elsevier, Amsterdam (1988)
17. Reinelt, G.: TSPLIB. University of Heidelberg (1996), <http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/>
18. Simões, A., Costa, E.: An immune system-based genetic algorithm to deal with dynamic environments: diversity and memory. In: Proc. of the 6th Int. Conf. on Neural Networks and Genetic Algs., pp. 168–174 (2003)
19. Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Proc. of the 2005 Genetic and Evol. Comput. Conf., pp. 1115–1122 (2005)
20. Yang, S.: A comparative study of immune system based genetic algorithms in dynamic environments. In: Proc. of the 2006 Genetic and Evol. Comput. Conf., pp. 1377–1384 (2006)
21. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)
22. Zhou, A., Kang, L., Yan, Z.: Solving dynamic TSP with evolutionary approach in real time. In: Proc. of the 2003 IEEE Congr. on Evol. Comput., pp. 951–957 (2003)

Dynamic Time-Linkage Problems Revisited

Trung Thanh Nguyen and Xin Yao

The Centre of Excellence for Research in Computational Intelligence and Applications
(CERCIA), School of Computer Science,
University of Birmingham, B15 2TT, United Kingdom
{T.T.Nguyen,X.Yao}@cs.bham.ac.uk

Abstract. Dynamic time-linkage problems (DTPs) are common types of dynamic optimization problems where "decisions that are made now ... may influence the maximum score that can be obtained in the future"[3]. This paper contributes to understanding the questions of what are the unknown characteristic of DTPs and how to characterize DTPs. Firstly, based on existing definitions we will introduce a more detailed definition to help characterize DTPs. Secondly, although it is believed that DTPs can be solved to optimality with a perfect prediction method to predict function values [3] [4], in this paper we will discuss a new class of DTPs where even with such a perfect prediction method algorithms might still be deceived and hence will not be able to get the optimal results. We will also propose a benchmark problem to study that particular type of time-linkage problems.

1 Introduction

This paper studies some unknown characteristics and the solvability of some classes of dynamic time-linkage problems (DTP)¹. DTP is a common type of dynamic optimization problems (DOPs) in both real-world combinatorial (scheduling [6], vehicle routing [8], inventory management [4]) and continuous domains (non-linear predictive control [13], optimal control theory[7]) but has not yet received enough attention from the Evolutionary Algorithms research. They are defined as problems where "...there exists at least one time $0 \leq t \leq t^{end}$ for which the dynamic optimization value at time t is dependent on at least one earlier solution..." [5].

Although the importance of DTPs have been shown through their presence in a broad range of real-world applications, due to the lack of research attention there are still many characteristics that we do not fully know about this type of problems. For example, how should we define and classify DTPs in detail; is there any characteristics of DTPs that we do not know; with these characteristics are DTPs still solvable; and what is the appropriate strategy to solve them.

This paper contributes to the work on finding partial answers for the above questions. Firstly, based on existing definitions we will introduce a more detailed

¹ "Dynamic time-linkage problem" is the term given by Bosman[3]. In other fields this type of problems is given different names, for example "model predictive control" or "anticipative decision process" although the nature of the problems is the same.

definition to help characterize DTPs and other DOPs. Secondly, although it is believed that DTPs can be solved to optimality with a perfect prediction method to predict future function values [3] [4], in this paper we will discuss a new class of DTPs where even with a perfect prediction method algorithms might still be deceived and hence will not be able to get the optimal results. We will also propose a benchmark problem to study that particular type of DTPs.

2 Existing Definition of Dynamic Time-Linkage Problems

A DTP is firstly a dynamic optimization problem, hence it also has all the characteristics of a regular DOP. The additional feature of a DTP, which makes it different from normal DOPs, is that the dynamic of a parameter may depend not only on the time variable, but also on earlier decisions made by the algorithm. It means that at the current time t^{now} the value of the parameter $\gamma(t^{now})$ of a function f may depend on the value of the variable $x(t)$, $0 \leq t < t^{now}$ found by the algorithm at at least one point before t^{now} .

Equation 1 shows the formal definition (proposed in [3]) for DOPs (including DTPs) with the time variable $t \in \mathbb{T} = [0, t^{end}], t^{end} > 0$.

$$\begin{aligned} & \max \{F_\gamma(x(t))\} \text{ subject to } C_\gamma(x(t)) = \text{feasible with} \\ & F_\gamma(x(t)) = \int_0^{t^{end}} f_{\gamma(t)}(x(t)) dt \\ & C_\gamma(x(t)) = \begin{cases} \text{feasible} & \text{if } \forall t \in [0, t^{end}] : C_{\gamma(t)}(x(t)) = \text{feasible} \\ \text{infeasible} & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

where γ are the time-dependent parameters of f and C is the constraint function.

Although the definition above is useful to generalize DTPs, it might not be detailed enough if we want to characterize one important property of DTP instances: *algorithm-dependency*. We consider DTP instances algorithm-dependent because the structure of a DTP in the future may depend on the current value of $x(t)$, which in turn depends on the algorithm used to solve the problem. At a particular change step t , different algorithms might provide different solutions $x(t)$, hence changing the future problem in different ways. Because of this property, we believe that in order to define a DTP in an unambiguous way, the algorithm used to solve a problem instance should be considered a part of that instance. The original definition in eq. 1 does not fully encapsulate this.

Another reason for us to formulate an extended definition is that in (eq. 1) the time-linkage feature is not explicitly expressed. Instead, that feature is encapsulated in the expression of $f_{\gamma(t)}$. It would be better if the time-linkage property can be captured explicitly in the definition. This has been partially done in [5] and here we will extend that concept further by including previous solutions that affect future function values in the definition.

In addition, because a DTP is firstly a dynamic problem, it would be useful if its definition is formulated in a detailed enough level to cover all characteristics of a DOP. Such a detailed definition would help to answer the question of (1) how to classify/characterize DTPs effectively and (2) how to separate the difficulty caused by the underlying static problem from the difficulty caused by

the dynamic/time-linkage feature. To provide such a detailed level, the proposed definition will cover such aspects of a DOP as how to control changes, how to express changes in the domain range/constraints, how to control change frequency and how to integrate the frequency of change into problem descriptions. Some aspects as time unit [1], [12], controlling changes [12] [9], and changes in constraints [3], [5] have been mentioned elsewhere. In this paper these aspects will be described in a more formal and detailed level. Other aspects are introduced for the first time in this paper.

3 A More Detailed Definition for DTPs

Definition 1 (Optimization algorithm and its solutions). *Given a dynamic problem f_t at the time step t and a set P_t of k_t point $\mathbf{x}_1, \dots, \mathbf{x}_{k_t} \in S_t$ where $S_t \subseteq \mathbb{R}^n$ is the search space², an optimization algorithm G can be seen as a mapping: $G_t : \mathbb{R}^{n \times k_t} \rightarrow \mathbb{R}^{n \times k_{t+1}}$ capable of producing a solution set P_{t+1} of k_{t+1} optimised points $\mathbf{x}_1^G, \dots, \mathbf{x}_{k_{t+1}}^G$ at the time step³ $t + 1$: $P_{t+1} = G_t(P_t)$.⁴ Generally, at a time step $t^e \in \mathbb{N}^+$ the set of solutions that we get by applying an algorithm G to solve f_t with a given initial populations P_{t^b-1} during the period*

$$[t^b, t^e], t^b \geq 1, \text{ is denoted } X_{f_t}^{G[t^b, t^e]} = \bigcup_{t=t^b}^{t^e} P_t = \bigcup_{t=t^b}^{t^e} G_t(P_{t-1})$$

Definition 2 (Full-description form). *Given a mathematical expression $\hat{f}_\gamma(x)$ with a variable x and a set of parameters $\gamma \in \mathbb{R}^m$, we call $\hat{f}_\gamma(x)$ the full-description form of a set of mathematical expressions $\{f_1(x), \dots, f_n(x)\}$ if there exists a set of vectors $\{\mathbf{c}_1, \dots, \mathbf{c}_n\}, \mathbf{c}_i \in \mathbb{R}^m, i = 1 : n$ so that if $\gamma = \mathbf{c}_i$ then $\hat{f}_{\gamma=\mathbf{c}_i}(x)$ is equal to $f_i(x)$. In other words:*

$$\begin{aligned} \hat{f}_\gamma(x) &\stackrel{\gamma=\mathbf{c}_1}{\rightarrow} f_1(x) \\ &\dots \\ &\stackrel{\gamma=\mathbf{c}_n}{\rightarrow} f_n(x) \end{aligned} \tag{2}$$

Each function $f_i(x), i = 1 : n \in N^+$ is called an instance of the full-description form $\hat{f}_\gamma(x)$ at $\gamma = \mathbf{c}_i$. \square

Example 1. The expression $ax + b$ is the full-description form of a set of expressions $\{f_1 = x; f_2 = 1; f_3 = x + 1\}$ with respect to the following set of values for a and b : $\{\{a = 1, b = 0\}, \{a = 0, b = 1\}, \{a = 1, b = 1\}\}$.

In the original definition (eq. 1), it is implied that changes in dynamic problems can be represented as changes in the parameter space. This has been described

² Here we are considering search spaces $\subseteq \mathbb{R}^n$ but it can be generalized for non-numerical encoding algorithms by replacing \mathbb{R}^n with the appropriate encoding space.

³ As mentioned in [1] and [12], from the perspective of optimization algorithms time is discrete and the smallest time unit is one function evaluation.

⁴ To some extents the mapping G is similar to the *generation transition function* (GTF) used to represents the EA population sequence in [2]. However, different from GTF, in this paper G is used to represent any type of optimization algorithms.

more explicitly in [12] and implemented by us in [9]. In this paper this concept will be formulated in a more formal level and will be explicitly made applicable to DTPs: *most common types of changes in dynamic (time-linkage) problems can be represented as changes in the parameter space if we can formulate the problem in a general enough full-description form.* For example, a function-switching change from $f(x)$ at $t = 0$ to $g(x)$ at $t \geq 1$, $t \in \mathbb{N}^+$ can be expressed as $\hat{f}(x) = a(t)f(x) + b(t)g(x)$ where $a(t)$ and $b(t)$ are two time-dependent parameters given by $\begin{cases} a(t) = 1 \text{ and } b(t) = 0 \text{ if } t = 0 \\ a(t) = 0 \text{ and } b(t) = 1 \text{ otherwise} \end{cases}$

In real-world problems, changes in the parameters are usually controlled by some specific time-dependent rules or functions⁵. Some time-dependent rules may also have the time-linkage feature, i.e. they also take solutions found by the algorithm up to the current time step as their parameters. The dynamic rules (non-time-linkage and time-linkage) can be defined mathematically as follows.

Definition 3 (Time-linkage dynamic driver). Given a tuple $\langle t, \gamma_t, X_{\hat{f}}^{G[1,t]} \rangle$ where $t \in \mathbb{N}^+$ is a change step variable; $\gamma_t \in \mathbb{R}^m$ is an m -element vector containing all m time-dependant parameters of a full-description form \hat{f} at t ; and $X_{\hat{f}}^{G[1,t]}$ is a set of k n -dimensional solutions achieved by applying an algorithm G to solve \hat{f} during the period $[1, t]$; a dynamic rule $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$ can be seen as a mapping $\mathbb{R}^m \times \mathbb{R}^{n \times k} \times \mathbb{N}^+ \rightarrow \mathbb{R}^m$. The output of $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$ is a vector $\gamma_{t+1} \in \mathbb{R}^m$

$$\gamma_{t+1} = D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t) \quad (3)$$

If all elements of γ_{t+1} are used as values for m time-dependant parameters of \hat{f} at the next change step $t + 1$ then we call $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$ the time-linkage dynamic driver⁶ of the time-dependant parameters set γ_t of \hat{f} . \square

There are cases where $X_{\hat{f}}^{G[1,t]}$ does not have any influence on the future of \hat{f} .

In these cases $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$ becomes a regular dynamic driver with no time-linkage feature.

Definition 4 (Time unit). In a dynamic optimization problem, a time unit, or a unit for measuring time periods in the problem, represents the time durations

⁵ For example, in some real-world systems changes of parameters can be represented by a linear, chaotic or other non-linear equations of the time variable t . Dimensional changes can also be represented as changes in the parameter given that the maximum number of variables is taken into account in the full-description form. For example, the function $\sum_{i=1}^n x_i^2$ with dimension n varies from 1 to 2 can be represented as the full-description form $\sum_{i=1}^2 b_i(t)x_i^2$ with $b_i(t) \in \{0, 1\}$ depending on t .

⁶ *Dynamic driver* is a term used by Morrison[10] in his work to refer to the logistic function in his benchmark. We can borrow this term to refer to any rules/functions that control the dynamic of a dynamic problem.

needed to complete one function evaluation of that problem.⁷ The number of evaluations (or time units) that have been evaluated so far in a DOP is measured by the variable $\tau \in N^+$.

Definition 5 (Change step and change frequency). In a DOP, a change step represents the moment when an environmental change happens. The number of change steps that have happened so far in a dynamic environment is measured by the variable $t \in N^+$. Obviously t is a time-dependent function of τ - the number of evaluations made so far; $t(\tau) : N^+ \longrightarrow N^+$. Because $t(\tau)$ is also a time-dependent parameter of the DOP f , its dynamic is controlled by a problem-specific time-based dynamic driver $D_T(t(\tau), \tau)$. D_T decides the frequency of change of the problem and can be described as follows:

$$\begin{cases} t(1) = 1 \\ t(\tau + 1) = t(\tau) + 1 \text{ if } D_T(t(\tau), \tau) = \text{true} \\ t(\tau + 1) = t(\tau) \quad \text{otherwise} \end{cases} \quad (4)$$

Definition 6 (Dynamic time-linkage optimization problem). Given a tuple $\langle \hat{f}, \hat{C}, D_P, D_D, D_T, G \rangle$, a dynamic time-linkage optimization problem in the period $[1, \tau^{end}]$ function evaluations, $\tau^{end} \in N^+$, can be defined as

$$\max \left\{ \sum_{\tau=1}^{\tau^{end}} \hat{f}_{\gamma(t_\tau, X_{\hat{f}}^{G[1,t]})}(\mathbf{x}_t) \right\} \quad (5)$$

subject to $\hat{C}^{i=1:k \in N^+}_{\gamma(t_\tau, X_{\hat{f}}^{G[1,t]})}(\mathbf{x}_t, t_\tau) \leq 0$; and $\mathbf{l}(t_\tau, X_{\hat{f}}^{G[1,t]}) \leq \mathbf{x}_t \leq \mathbf{u}(t_\tau, X_{\hat{f}}^{G[1,t]})$

where

\hat{f} is the full-description form of the objective function

$\hat{C}^1 \dots \hat{C}^k$ are the full-description forms of k dynamic constraints⁸

D_P is the dynamic driver for parameters in objective and constraint functions (see below)

D_D is the dynamic driver for domain constraints (see below)

D_T is the dynamic driver for times and frequency of changes(eq. 4)

G is the algorithm used to solve the problem

$\tau \in [1, \tau^{end}] \cap N$ is the number of function evaluations done so far

t_τ , or $t(\tau) \in N^+$ is the current change step; $t(\tau)$ is controlled by D_T (eq. 4)

$X_{\hat{f}}^{G[1,t]}$ is the set of solutions achieved by applying the algorithm G to solve \hat{f} during $[1, t]$

$\gamma_{t_\tau} \in \mathbb{R}^p$ is the time-dependant parameters of \hat{f} and \hat{C}^i ; $\gamma_{t_\tau+1} =$

$D_P(\gamma_{t_\tau}, X_{\hat{f}}^{G[1,t]}, t)$

$\mathbf{l}(t_\tau), \mathbf{u}(t_\tau) \in \mathbb{R}^n$ are domain constraints; $\begin{cases} \mathbf{l}(t_\tau+1) = D_D(\mathbf{l}(t_\tau), X_{\hat{f}}^{G[1,t]}, t_\tau) \\ \mathbf{u}(t_\tau+1) = D_D(\mathbf{u}(t_\tau), X_{\hat{f}}^{G[1,t]}, t_\tau) \end{cases}$

⁷ This has been mentioned in [1] and [12].

The new definition brings us some advantages. Firstly, we can now classify DTPs based on three distinguished components: the full-description forms (determine the statics), the dynamic drivers (determine the dynamics), and the solvers (determine the future problem). Secondly, it supports an important yet not fully considered feature of DTP instances: algorithm-dependency. Finally it represents a general case when solutions from multiple previous steps can affect future performance. The definition can also be used to represent other types of DOPs.

4 Time-Deceptive and the Anticipation Approach

According to [5], a dynamic problem is said to be time-deceptive toward an optimizer if the problem is time-linkage and the optimizer cannot efficiently take into account this time-linkage feature during its optimization process.

Bosman[3] illustrates this property by proposing the following benchmark:

$$\text{given } n = 1; h(x) = e^x - 1; \max_{x(t)} \left\{ \int_0^{t^{end}} f(x(t), t) dt \right\} \quad (6)$$

$$f(x_t, t) = \begin{cases} -\sum_{i=1}^n (x(t)_i - t)^2 & \text{if } 0 \leq t < 1 \\ -\sum_{i=1}^n [(x(t)_i - t)^2 + h(|x(t-1)_i|)] & \text{otherwise} \end{cases}$$

The benchmark problem above is a DTP because for any $t \geq 1$, the current value of $f(x, t)$ depends on $x(t-1)$ found at the previous time step.

An interesting property is revealed when we try to optimize the above problem using the traditional approach: optimizing the present. That property is: *the trajectory formed by optimum solutions at each time step might not be the optimal trajectory*. For example, in figure 1 we can see that the trajectory of $f(x^*, t)$ when we optimize the present (with optimum solution $\mathbf{x}^*(t) = t$ at the time step t) is actually worse than the trajectory of a $f(x^0, t)$ with a simple solution $\mathbf{x}^0 = 0 \forall t$. It means that the problem is deceptive because an optimizer following the traditional approach is not able to take into account the time-linkage feature.

Bosman [3] [5] suggested that DTPs can be solved to optimality by estimating the values of the function for future times given a trajectory $\cup_{t=0}^{t^{now}} \{f_t, t\}$ of history data and other previously evaluated solutions. From that estimation, we can choose a future trajectory with optimal future function values. In other words, it is suggested that time-linkage problems can be "solved to optimality" by prediction methods and the result could be "arbitrarily good" if we have a "perfect predictor" [3] [4] [5]⁹. The authors also made some experiments on the benchmark problem mentioned in eq. 6 and on the dynamic pickup problem, showing that under certain circumstances prediction methods do help to improve the performance of the tested algorithms.

⁸ These also include equality constraints because any equality constraint $c(x) = 0$ can be transformed into an inequality $|c(x)| - \varepsilon \leq 0$ with a small value ε .

⁹ A predictor, as defined in [5, line 8-12, pg 139], is "a learning algorithm that approximates either the optimization function directly or several of its parameters... When called upon, the predictor returns either the predicted function value directly or predicted values for parameters". Hence, perfect predictors should be ones that can predict values exactly as the targets.

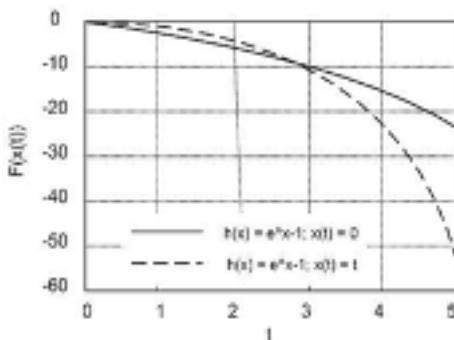


Fig. 1. This figure (reproduced from [3]) illustrates the time-deception property. We can see that the trajectory of $f(x_t)$ when we optimize the present (dash line, with optimum solution $x(t) = t$) is actually worse than the trajectory of $f(x_t)$ with a simple solution $x(t) = 0$ (the solid line). To solve this problem to optimality, we need to use a predictor to predict the trajectory of function values given different outcomes of current solutions, then choose the one that give us the maximum profit in the future.

5 Can Anticipation Approaches Solve All DTPs?

Contrary to existing belief, we will show below that there might be cases where the hypothesis above does not hold: even if we use a "perfect predictor" to predict future function values, if during the predicted time span, the trajectory of the future function values changes its function form, it might not be possible to solve the time-linkage problems to optimality.

Let us consider the situation where predictors help achieving optimal results first. At the current time $t^{now} \geq 1$, in order to predict the values of $f(x(t))$ at a future time t^{pred} , a predictor needs to take the history data, for example the previous trajectory of function values $Z^{[0,t^{now}-1]} = \cup_{t=0}^{t^{now}-1} \{f_t, t\}$, as its input. Given that input, a perfect predictor would be able to approximate correctly the function form of $Z^{[0,t^{now}-1]}$ and hence would be able to predict precisely the future trajectory $Z^{[t^{now}, t^{pred}]}$ if it has the same function form as $Z^{[0,t^{now}-1]}$.

One example where predictors work is the problem in eq. 6. In that problem, for each trajectory of $x(t)$ the trajectory of $f(x(t))$ always remains the same. For example with $x(t) = t$ the trajectory is always $1 - e^{t-1}$ or with $x(t) = 0$ the trajectory is always $-t^2$ (see figure 1). As a result, that problem is predictable.

Now let us consider a different situation. If at any particular time step $t^s \in [t^{now}, t^{pred}]$, the function form of $Z^{[t^{now}, t^{pred}]}$ changes, the predicted trajectory made at t^{now} to predict $f(x(t))$ at t^{pred} is no longer correct. This is because before t^s there is no information about how the the function form of $Z^{[t^{now}, t^{pred}]}$ would change. Without such information, it is impossible to predict the optimal trajectory of function values after the switch, regardless of how good the predictor is. It means that the problem cannot be solved to optimality even if we use a perfect predictor to predict function/parameter values.

To illustrate this situation, let's consider the following simple problem where the trajectory of function values changes over time (illustrated in figure 2).

$$\widehat{F}(x_t) = a_t f(x_t) + b_t g(x_t) + c_t h(x_t) \quad 0 \leq x_t \leq 1 \quad (7)$$

where $\widehat{F}(x)$ is the full-description form of a dynamic function; $f(x_t) = x_t$; $g(x_t) = x_t + (d - 2)$; $h(x_t) = x_t + d$; a_t, b_t and c_t are the time-dependent parameters of $\widehat{F}(x_t)$. Their dynamic drivers are set out as follows:

$$\begin{cases} a_t = 1; b_t = c_t = 0 & \text{if } (t < t^s) \\ a_t = 0; b_t = 1; c_t = 0 & \text{if } (t \geq t^s) \text{ and } (\widehat{F}_{t^s-1}(x_{t^s-1}^G) \geq 1) \\ a_t = 0; b_t = 0; c_t = 1 & \text{if } (t \geq t^s) \text{ and } (\widehat{F}_{t^s-1}(x_{t^s-1}^G) < 1) \end{cases} \quad (8)$$

where $t^s > 1$ is a pre-defined time step, $d \in \mathbb{R}$ is a pre-defined constant, and $x_{t^s-1}^G$ is a single solution produced at $t^s - 1$ by an algorithm G . Eq. 8 means that with $t < t^s$, the form of $\widehat{F}(x_t)$ is always equal to $f(x_t)$; with $t \geq t^s$, depending on the solution of $x_{t^s-1}^G$ the form of $\widehat{F}(x_t)$ would switch to either $g(x_t)$ or $h(x_t)$.

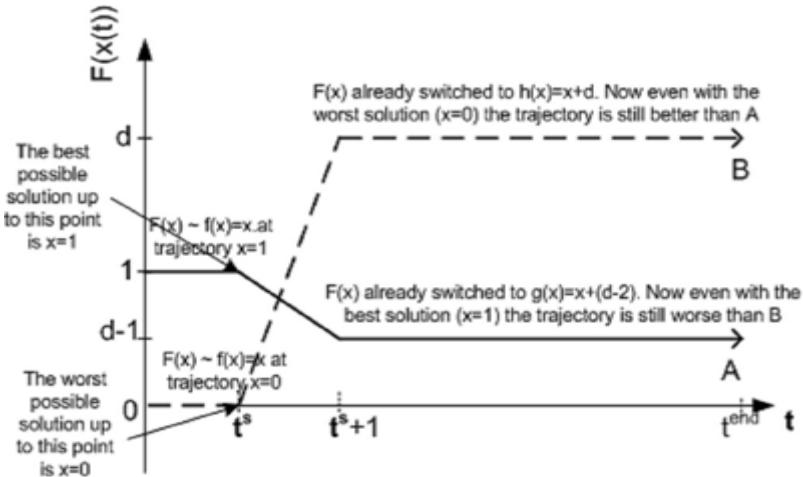


Fig. 2. This figure illustrates a situation where even the best predictor + the best algorithm (A) still perform worse than the worst predictor + the worst algorithm (B) due to the prediction-deceptive property of the problem in eq.7. Assume that we want to predict the trajectory of $F(x)$ from $[0, t^{end}]$. In case A, the best predictor allows us to predict $F(x) \sim f(x) = x$ in just only one time step $[0, 1]$. With that perfect prediction the algorithm is able to find the best solution $x = 1$, which is valid until t^s . Now at t^s although the history data tells the predictor that the trajectory must still be $F(x) \sim f(x) = x$, according to eq.8 the actual $F(x)$ does switch to $g(x) = x + (d - 2)$, which is the worst trajectory. In other words, the best predictor chose the worst trajectory to follow. On the contrary, in the case B the worst predictor+worst algorithm actually get benefit from the switch: the terrible solution ($x = 0$) they found during $[0, t^s]$ does help them to switch to $F(x) \sim h(x) = d + x$, whose trajectory after t^s is always better than A regardless of the value of x .

In the above problem, because at any $t \geq t^s$ the values of a_t , b_t and c_t (and consequently the value of the function \hat{F}) depend on the solution found by G at $t^s - 1$, according to the definition in [5] the problem is considered time-linkage.

This problem has a special property: at any $t < t^s$ one can only predict the value of \hat{F} up to $t^s - 1$. Before t^s , history data does not reveal any clue about the switching rule in eq. 8, hence it is impossible to predict (1) whether the function will switch at t^s ; (2) which value $x_{t^s-1}^G$ should get to switch $\hat{F}(x_t)$ to $g(x_t)$ / $h(x_t)$ and (3) which form, g or h , would provide better future trajectory.

Even worse, even a perfect predictor might still be deceived to provide worse result than not using any predictor while solving this time-linkage problem! Figure 2 illustrates the situations where the best predictor could provide the worst result while the worst predictor could provide better results after t^s !

Summarizing, the example problem we proposed in this section illustrates a previously unknown class of DTPs where it is not guaranteed to get optimal results even if we have a perfect predictor to predict the function values. We call this class *time-linkage problems with unpredictable optimal function trajectories*. The example illustrates a special case where the predictor can be deceived and hence provide the worse results than not using predictor at certain time steps. We call these types of problems the *prediction-deceptive time-linkage problems*.

6 Another Benchmark for Time-Linkage Problems

Currently the only available DTP benchmarks are from [3][5]. Both unfortunately do not cover the prediction-deceptive case. To study prediction-deception and other types of DTPs, besides the problem that we have proposed in eq.7, we also developed a more comprehensive benchmark which covers all three types of dynamics: non-time-linkage, predictable time-linkage and prediction-deceptive time-linkage. Due to limited space we do not introduce details of the benchmark in this paper. Interested readers are referred to the technical report in [11].

7 Conclusion

In this paper we have proposed an extended definition for DTPs/DOPs where the problems can be classified based on three components: the full-description forms (determine the static description), the dynamic drivers (determine the dynamics), and the solver (determine how the future problem would be). It is expected that by separating the dynamics/time-linkage feature from the static part of the problem, the definition would make it easier to study the behaviour of DTPs. The definition will also help in generating DTP benchmarks from existing well-studied static benchmarks. It also takes into account aspects as time unit, change step, change frequencies and domain range changes. These details help defining, classifying and characterizing DTPs better.

Secondly, the paper revealed a new class of DTPs that has not been considered before: DTPs with unpredictable optimal function trajectories. We have illustrated that, contrary to previous suggestions, this class of DTPs cannot be solved to optimality using the traditional anticipation approaches.

Finally, we have proposed a time-linkage benchmark problem to study that particular type of time-linkage problems. We also provided a link to a technical report where a more complete set of DTP benchmarks are described.

Acknowledgement

The authors are grateful to P. Rohlfschagen, T. Ray, L. Xing and three anonymous reviewers for their helpful comments. This work is partially supported by an EPSRC grant (No. EP/E058884/1) on "Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis & Applications", an ORS Award and a School of Computer Science PhD Studentship.

References

1. Bäck, T.: On the behavior of evolutionary algorithms in dynamic environments. In: Proc. of the IEEE Intl. Conf. on Evolutionary Computation, pp. 446–451 (1998)
2. Bäck, T.: *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford Univ. Press, Oxford (1996)
3. Bosman, P.A.N.: Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Proc. of the GECCO 2005 Conf., pp. 39–47 (2005)
4. Bosman, P.A.N., Poutré, H.L.: Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In: GECCO 2007: Proc. of the 9th Conf. on Genetic and Evolutionary Computation, pp. 1165–1172. ACM, New York (2007)
5. Bosman, P.A.N.: Learning and Anticipation in Online Dynamic Optimization. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) *Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 129–152. Springer, Berlin (2007)
6. Branke, J., Mattfeld, D.: Anticipation in dynamic optimization: The scheduling case. In: Deb, K., et al. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 253–262. Springer, Heidelberg (2000)
7. Dorfman, R.: An Economic Interpretation of Optimal Control Theory. *American Economic Review* 59(5), 817–831 (1969)
8. van Hemert, J.I., La Poutre, J.A.: Dynamic routing problems with fruitful regions: Models and evolutionary computation. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 692–701. Springer, Heidelberg (2004)
9. Li, C., Yang, S., Nguyen, T.T., Yu, E.L., Yao, X., Jin, Y., Beyer, H.-G., Suganthan, P.N.: Benchmark Generator for CEC 2009 Competition on Dynamic Optimization, Technical report, University of Leicester and University of Birmingham, UK (2008), <http://www.cs.le.ac.uk/people/syang/ECiDUE DBG.tar.gz>
10. Morrison, R.W.: *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, Berlin (2004)
11. Nguyen, T.T.: A benchmark for dynamic time-linkage problems, Technical Report, School of Computer Science, University of Birmingham (2009)
12. Rohlfschagen, P., Yao, X.: Attributes of combinatorial optimisation. In: Proc. of the 7th Int. Conf. on Simulated Evolution and Learning, pp. 442–451. Springer, Heidelberg (2008)
13. Sistu, P.B., Gopinath, R.S., Bequette, B.W.: Computational issues in nonlinear predictive control. *Comput. Chem. Eng.* 17, 361–367 (1993)

The Dynamic Knapsack Problem Revisited: A New Benchmark Problem for Dynamic Combinatorial Optimisation

Philipp Rohlfshagen and Xin Yao

The Centre of Excellence for Research in Computational
Intelligence and Applications (CERCIA)
School of Computer Science, University of Birmingham
Birmingham B15 2TT, United Kingdom
P.Rohlfshagen@cs.bham.ac.uk, X.Yao@cs.bham.ac.uk

Abstract. In this paper we propose a new benchmark problem for dynamic combinatorial optimisation. Unlike most previous benchmarks, we focus primarily on the underlying dynamics of the problem and consider the distances between successive global optima only as an emergent property of those dynamics. The benchmark problem is based upon a class of difficult instances of the 0/1-knapsack problem that are generated using a small set of real-valued parameters. These parameters are subsequently varied over time by some set of difference equations: It is possible to model approximately different types of transitions by controlling the shape and degree of interactions between the trajectories of the parameters. We conduct a set of experiments to highlight some of the intrinsic properties of this benchmark problem and find it not only to be challenging but also more representative of real-world scenarios than previous benchmarks in the field. The attributes of this benchmark also highlight some important properties of dynamic optimisation problems in general that may be used to advance our understanding of the relationship between the underlying dynamics of a problem and their manifestation in the search space over time.

1 Introduction

The field of evolutionary computation (EC) provides a promising framework under which numerous competitive algorithms have been developed to address a wide range of (stationary) *NP-hard* optimisation problems. Lately, an increasing amount of attention has been paid to dynamic optimisation problems (DOPs), where the specification of the problem changes over time. Such problems are generally regarded as more representative of real-world scenarios where uncertainties are commonplace. Population-based evolutionary algorithms (EAs) are expected to perform well in the dynamic domain as evolutionary dynamics in nature also take place in a highly uncertain environment. It therefore comes as no surprise that numerous new EAs have been suggested in recent years to tackle the issue of time-dependent fitness landscapes.

The field of dynamic optimisation ([2,12,25,7]) is relatively new and although some early work dates back more than 40 years, it was only in the last 15-20 years that the field gained significant momentum. One of the earliest problems considered was a dynamic variant of the 0/1-knapsack problem (KP), a widely studied combinatorial optimisation problem that has several direct counterparts in industry, such as the cutting stock problem or resource allocation in distributed systems. Its objective is to fill a knapsack of capacity c with any number of n items, each with weight w_i and profit p_i , in such a way that the combined profits of all items is maximized without exceeding the knapsack's capacity. More formally, the aim is to optimise the fitness

$$\max \left\{ \sum_{i=1}^n p_i x_i \right\} \quad \text{w.r.t.} \quad \sum_{i=1}^n w_i x_i \leq c \quad (1)$$

where $x_i \in \{0, 1\}$. In [6], a $n = 17$ KP instance was proposed where the capacity c varies between two pre-determined values (i.e., the capacity, now denoted as $c(t)$, is time-dependent). This particular instance has been used in numerous other papers (e.g., [24,20,19]) while additional work has focused on larger instances of the same problem (e.g., [10,9]). Furthermore, numerous publications deal with the more constrained multiple-knapsack problem (MKP).

The MKP is a generalisation of the single KP with m knapsacks, each of capacity c_j . The weight of each item depends on the knapsack such that the goal of the MKP is defined as

$$\max \left\{ \sum_{i=1}^n p_i x_i \right\} \quad \text{w.r.t.} \quad \sum_{i=1}^n w_{ij} x_i \leq c_j, \quad \forall j \quad (2)$$

where $x_i \in \{0, 1\}$. In a previous dynamic version of the MKP, all parameters (i.e., weights, profits and constraints) have been made dynamic (e.g., [4]) using a normally distributed random distribution with zero mean and standard deviation θ :

$$p_i^+ = p_i(1 + N(0, \theta_p)), \quad w_{ij}^+ = w_{ij}(1 + N(0, \theta_w)), \quad c_j^+ = c_j(1 + N(0, \theta_c)) \quad (3)$$

We believe the KP¹ to be an excellent benchmark for testing novel algorithms in the dynamic domain. The reasons for this are as follows:

1. The KP has many real-world counterparts (e.g., encryption, cutting stock) and is thus of industrial interest.
2. The KP, like most real-world problems, is constrained, an aspect often ignored in dynamic continuous benchmark problems (e.g., [2,12]).
3. Numerous publications have already addressed and analysed both the static and, to a limited degree, dynamic version of this particular problem.

¹ We also believe the MKP to be a promising benchmark problem for work in dynamic optimisation although here we only focus on the simpler KP.

4. A binary representation is a natural choice for KP, one of the most commonly used and studied encodings in the GA literature.
5. Although the KP is considered to be one of the “easier” NP-hard problems, it is possible to systematically generate instances which are unlikely to be solved efficiently by state-of-the-art algorithms [15].
6. Numerous studies of different penalty functions and repair algorithms already exist in the literature (e.g., [14,16]).
7. The role of the representation (binary and otherwise) has been analysed in both the static and dynamic case (e.g., [21,3]).

Many existing dynamic benchmark problems (see section 2.2) focus on simulating a wide range of different dynamics to provide a comprehensive framework under which different algorithms may be compared. These benchmarks have recently been criticised by Ursem et al. as “no research has been conducted to thoroughly evaluate how well they reflect characteristic dynamics of real-world problems.” [23, p 1]. The authors further note that such benchmarks focus only on how the landscape changes over time and not the underlying dynamics themselves. More specifically, however, almost all attention is paid to changes to the global optimum and this focus may divert attention away from dynamics taking place elsewhere in the search space. Furthermore, the distance between successive global optima, although important, does not necessarily imply much about the difficulty of the transition. The structure of the search space connecting successive global optima is likely to be equally important. In this paper, we focus primarily on the underlying dynamics and how they affect the parameters of the problem. We also focus on the movement of the global optimum, as it seems the only available metric to measure the relatedness successive problem instances, but only as an emergent property of the well-defined underlying dynamics.

2 Dynamic Optimisation Problems

DOPs come in many different variations but here we only consider the simplest case as it happens to be the most frequently studied one in the literature: The values of the parameters of the problem change over time with periods of inactivity (stagnation) between changes. This type of DOP may be viewed as a time-series of instances of the same problem and it is hoped that one can outperform random restarts by “transferring knowledge from the past” [7]. The remainder of this section presents an extended problem definition and discusses existing benchmark problems that model this particular type of DOP.

2.1 Problem Definition

In [17], a dynamic combinatorial optimisation problem was defined using the notation originally proposed by Garey and Johnson [5, p 123]: A (stationary) combinatorial optimisation problem Π (either maximisation or minimisation) consists of a set S_Π of instances (parameters and constraints), a finite set of candidate solutions $X_\Pi(I)$ for each instance $I \in S_\Pi$ and a function $f_\Pi : I \times X \rightarrow \mathbb{R}$

that assigns a positive rational number $f_{\Pi}(I, \mathbf{x})$ to each pair of instance and candidate solution. The goal is to find, in the shortest time possible, the global optimum \mathbf{x}^* which corresponds to the element in X for which $f_{\Pi}(I, \mathbf{x}^*) \succ f_{\Pi}(I, \mathbf{x})$, $\forall_{\mathbf{x} \in X}$ where $\succ \in \{\leq, \geq\}$. The distinction between a problem Π and instances of said problem, S_{Π} , is crucial. The problem Π is an unambiguous description (mathematically or otherwise) of a problem and includes the definition of the objective function. Each instance contains the information (parameters and constraints) to actually compute the quality of a solution (decision variables). In static optimisation, one might be given an arbitrary function such as $f(x) = 2x^2$, $x \in [l, u]$, and is asked to find the maximum/minimum. In the dynamic case, one is given a *class* of functions described by the more general form $f(x; a, b) = ax^b$ and the algorithm has to follow the time-variant global optimum through the state space. In the following, we extend this definition to place even more emphasis on the underlying dynamics of the problem.

A DOP, Π_D , is specified not only by S_{Π} but also by the set of (dynamic) instances S_{Π_D} where each $\mathcal{T} \in S_{\Pi_D}$ is a mapping $\mathcal{T} : I \times \mathbb{N} \rightarrow I$, consisting of a set of difference equations:

$$\mathcal{T}_{\Pi}(I(T), T) = \begin{pmatrix} \mathcal{T}_{\Pi}^1(I_1(T), T) \\ \vdots \\ \mathcal{T}_{\Pi}^k(I_k(T), T) \end{pmatrix} \quad (4)$$

The dynamics encapsulated by \mathcal{T}_{Π} generate a new instance at $T + 1$, given the previous instance as well as the time itself: $I(T + 1) = \mathcal{T}(I(T), T)$ where T represents an environmental index such that

$$T = \begin{cases} T + 1 & \text{if } t \bmod \tau = 0 \\ T & \text{otherwise} \end{cases} \quad (5)$$

and where $1/\tau$ is the frequency of change (usually constant but could also be a time-variant function). The actual time t is assumed to be discrete and advances with every function evaluation. A dynamic optimisation problem may thus be specified by the bounded space of parameters (S_{Π}) and the space of difference equations S_{Π_D} which describe time-variant trajectories through S_{Π} . A particular instance of such a DOP is then given by the tuple $(\Pi, \mathcal{T}_{\Pi}, I(0))$ where $I(0)$ is the initial instance of the problem.

2.2 Existing Dynamic Benchmark Problems

The three most widely used benchmarks are due to Branke (MOVING PEAKS; [2]), Morrison (DF1; [12]) and Yang (XoR; [26]). The former two are continuous in nature and model the search space as a “field of cones” [13], each of which may be controlled individually to model different ranges of dynamics: The height, width and location of each peak is varied over time. In DF1, this is accomplished with the scaled output of a logistic function (see equation 6). The MOVING PEAKS benchmark is conceptually very similar and only differs in its

exact specifications. Numerous other continuous problems have been suggested, but none of them were developed explicitly as benchmark generators.

The popularity of continuous DOPs may, at least partly, be explained by the ability to fully specify the fitness landscape of a real-valued function $f(\mathbf{x})$. The fitness landscape metaphor is also applicable in the combinatorial domain although here it is only possible to model the fitness landscape in regard to the algorithm used (see [11,18,17]). XoR, a combinatorial benchmark, circumvents this issue and allows to generate a dynamic version of any static pseudo-boolean problem by rotating individual search points without altering the actual underlying function. This is done by performing a logical *xor* operation on each search point prior to each function evaluation such that the objective function becomes $f(\mathbf{x}(t) \oplus \mathbf{m}(T))$ where the vector \mathbf{m} is constructed systematically to model different magnitudes of change. More details about this problem may be found in [26,22]. Finally, it should be noted that in most scenarios, the frequency and magnitude of change are kept constant.

In this paper, we divert from the landscape-oriented design methodology and focus primarily on the underlying dynamics of the problem. We believe such undertaking to be essential to draw more attention to the combinatorial domain where a strict landscape-oriented model is usually not applicable: The dynamics affect the parameter space and resulting changes in successive fitness landscapes are *algorithm-dependent* and may, in the worst case, be entirely uncorrelated to the underlying rules that generate them [17]. Here we show that it is still possible to correlate the dynamics of the problem's parameters and the movement of the global optimum.

3 The Dynamic 0/1 Knapsack Problem

3.1 Dynamic Spanner Instances

In [15], Pisinger constructs numerous classes of KP instances where state-of-the-art algorithms either struggle or fail completely, even for relatively small number of dimensions (n). Here we consider the so-called *spanner* instances which are constructed from a small set of parameters known as the *spanner set*. Each such set is characterised by the pair (v, m) where v is the number of items in the spanner set and m is a multiplier limit. A set of v items is generated with weights in $[1, R]$ (i.e., R denotes the maximum possible weight) and profits according to some distribution. Here we chose *strongly correlated instances* such that for each $w_j \in [1, R]$, the corresponding profit is $p_j = w_j + R/10$. Finally, all items in the spanner set are normalised. The n items of the actual problem instance are then constructed by randomly choosing an item from the spanner set and a randomly chosen multiplier $a \in [1, m]$ such that each item is specified by $(a \cdot w_j, a \cdot p_j)$. It follows that the n items are split into v disjunctive sets, each with a different profit-weight ratio.

In order to make the problem dynamic, we denote the weight of spanner item j , w_j , as a fraction of the maximum possible weight, $\alpha_j R$, where $\alpha_j \in [0, 1]$. The weights and hence the profit-weight ratio of group j may then be altered by

changing the value of α_j . The dynamics are thus restricted to these coefficients and any type of function whose (scaled) output is in the range $[0, 1]$ may be used to describe the trajectory of each α_j . Numerous studies already exist where the dynamics modelled follow precise underlying rules. In the DF1 benchmark, for instance, Morrison makes use of the logistic map:

$$\alpha_j(t+1) = \mu\alpha_j(t)(1 - \alpha_j(t)) \quad (6)$$

This function, depending on μ and $\alpha_j(0)$ may produce a wide variety of different trajectories, from stationary points and repeating patterns to chaotic ones. In a similar fashion, Ursem et al. [23] consider a Fourier function to be used in a greenhouse benchmark problem:

$$f(t, A, a, B, b) = \sum_{i=1}^{|A|} a_i \cos(2\pi A_i t') + \sum_{i=1}^{|B|} b_i \sin(2\pi B_i t') \quad (7)$$

Here we do not consider a particular function as such but instead focus on more general properties of the problem: We are interested in the characteristics of the transitions from one instance to the next in relation to the trajectories of all α_j . Pisinger showed that a value of $v = 2$ seemed most difficult [15]. This simplifies the problem even further as now only two difference equations are required to model the dynamics of the problem.

3.2 Properties of the Problem

Each item is a multiple of one of the v spanner items with weight w_{α_i} and profit p_{α_i} . The profit-weight ratio of spanner item i depends on α_i and it follows that the lower α_i (and hence all items generated from this), the higher the profit-weight ratio (as we only consider strongly correlated items). Lower values of α_i also imply a lower expected weight of all items generated from this value². If we denote the set of all n items as $N = \{1, 2, \dots, n\}$, any solution to the KP is a subset of items $S \subseteq N$. The distance measure used in the experimental part of the paper is defined as the distance between two sets S_1 and S_2 :

$$d(S_1, S_2) = |(S_1 \cup S_2) \setminus (S_1 \cap S_2)| \quad (8)$$

We denote all items generated by α_i as set $N_{\alpha_i} \subseteq N$ such that $N_{\alpha_i} \cap N_{\alpha_j} = \emptyset, \forall j \neq i$. In the case of $v = 2$, the solution S may then be seen as the union of two subsets $S = S_{\alpha_1} \cup S_{\alpha_2}$ where $S_{\alpha_1} \subseteq N_{\alpha_1}$ and $S_{\alpha_2} \subseteq N_{\alpha_2}$. As one of the spanner sets has a higher profit-weight ratio as well as a lower expected weight (let us assume, for the sake of clarity, it is S_{α_1}), it is plausible to assume that $|S_{\alpha_1}| > |S_{\alpha_2}|$. Globally optimum solutions are thus, on average, largely made up of items from N_{α_1} with some items from N_{α_2} depending on the gap to the constraint c . This particular attribute allows one to approximately predict the distances between successive global optima which should be proportional to

² The expected sum of weights for all spanner items v_i is $E[\sum w_{\alpha_i}] = \alpha \cdot (R+1) \cdot m/2$.

the combined rate of change of all values α_i . In particular, we propose that given the two dynamic coefficients α_1 and α_2 , the approximate distance of successive global optima is

$$d(\mathbf{x}^*(T+1), \mathbf{x}^*(T)) \propto \frac{|[\alpha_1(T+1) - \alpha_1(T)] - [\alpha_2(T+1) - \alpha_2(T)]|}{2} \quad (9)$$

It is important to note that this relationship clearly depends on the value of c : If c is very large, for example, almost all items will be part of the global optimum at any one time and the maximum range of change severities is limited. Hence it is important to make c a time-variant function of the current weights and here we define $c(T)$ as:

$$c(T) = |\alpha_1(T) - \alpha_2(T)| \cdot \sum_{i=1}^{|N_{\alpha^*}(T)|} w_{N_{\alpha^*}(T)} \quad (10)$$

where $N_{\alpha^*}(T)$ is the spanner set with the superior profit-weight ratio (i.e., the lower value of α). This particular definition implies that, given the maximum possible difference between the two values of α , the global optimum is precisely equivalent $N_{\alpha^*}(T)$ (i.e. the global optimum is known). It should be noted that $c(T)$ is a function of α and does not introduce any additional parameters. In the remainder of this paper we demonstrate empirically that this relationship holds.

4 Experimental Setup

In the experiments we consider 30 time-series (trials) of problem instances that vary according to the chosen trajectories of α_1 and α_2 as shown in figure 1. Each such trial starts with a uniquely specified instance $I(0)$ and lasts for 40 changes. The general settings are identical to those used by Pisinger: $m = 10$, $v = 2$ and $R = 10^4$. Furthermore, we chose a value of $n = 20$ to allow for an exhaustive search to find the global optimum. For each experiment we look at

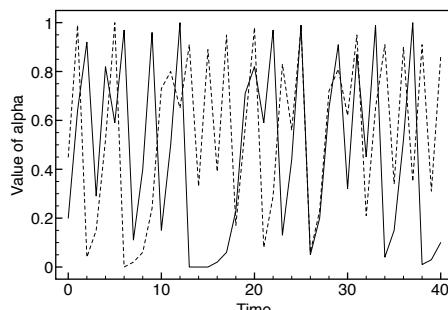


Fig. 1. Interactions of α_1 and α_2 over time: both trajectories are chaotic, generated by a logistic map (equation 6) using a value of $\mu = 4$

the distance from the global optimum at time T to the (closest) global optimum at time $T + 1$ and average over all trials. The trajectories chosen are chaotic as a representative sample over the different kinds of difference equations that could be used to model the dynamics (see equation 4).

5 Results

For each experiment, we look at the distances between all successive global optima $\mathbf{x}^*(T)$ to $\mathbf{x}^*(T + 1)$. The actual distances are compared to the estimated distances as shown in figure 2. It can be seen that the estimated distances are very close to the actual ones for all neighbouring points. The correlation between the combined rate of change of α_1 and α_2 and the distances between successive global optima is more clearly shown by figure 2 (right) and it is evident that there is a distinct relationship between the two values. It remains to be seen how well this relationship holds for other parameter settings (i.e., other values of R , v and m) but for now, this result may be used to construct a dynamic combinatorial benchmark problem with varying degrees of severity between changes, determined by the interaction of trajectories of α_1 and α_2 .

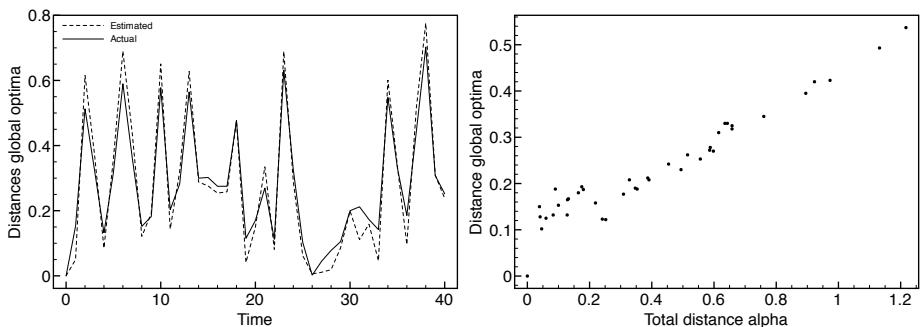


Fig. 2. Results: (left) the predicted and actual distances between successive global optima and (right) the correlation between the total differences in α_1 and α_2 and the actual distances between corresponding global optima

6 Conclusions and Future Work

We have shown how a well-known static optimisation problem may be turned into a dynamic equivalent with well-defined time-dependent underlying rules that govern the transition from one instance in time to the next. We concentrated on the knapsack problem (KP), a well-studied combinatorial optimisation problem with numerous real-world applications. In particular, we focused on a set of instances called *spanner instances* [15] which are known to be difficult for exact algorithms. Spanner instances are generated using a small set of well-defined parameters (the spanner set) such that the n items are split into v groups each of

which has a specific profit-weight ratio. This ratio depends on the initial weight chosen and here we control that weight over time, using a dynamic coefficient α , to model different kinds of dynamics. It is possible to model approximately different magnitudes of change (distances between successive global optima) by looking at the trajectories of the parameters even though the actual global optima are unknown. It has been shown that there is a direct relationship between the rate of change of the parameters and the distance of successive global optima.

In the near future we intend to address additional attributes such as time-linkage [1] to move closer to more realistic benchmark problems. It is also of interest to investigate how the dynamics of the problem affect the feasible area of the search space as this may severely affect the performance of the algorithm. Complimentary to the study carried out in [17], it would also be interesting to analyse the fitness distance correlations (FDCs; [8]) of the problem to determine whether distances of successive global optima are correlated to the FDCs of the underlying instances. Finally, it remains to be established whether the movement of the global optimum is proportional to the expected computational effort required by an algorithm to re-adapt once a change has taken place.

Acknowledgements

This work was supported by EPSRC grant no. EP/E058884/1 on “Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications”.

References

1. Bosman, P.A.N.: Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Proceedings of the 2005 workshops on Genetic and evolutionary computation, pp. 39–47 (2005)
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer, Dordrecht (2002)
3. Branke, J., Orbayi, M., Uyar, S.: The role of representations in dynamic knapsack problems. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) *EvoWorkshops 2006*. LNCS, vol. 3907, pp. 764–775. Springer, Heidelberg (2006)
4. Branke, J., Salihoglu, E., Uyar, S.: Towards an analysis of dynamic environments. In: Beyer, H.-G., et al. (eds.) *Genetic and Evolutionary Computation Conference*, pp. 1433–1439. ACM, New York (2005)
5. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York (1979)
6. Goldberg, D.E., Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Grefenstette, J.J. (ed.) *Second International Conference on Genetic Algorithms*, pp. 59–68. Lawrence Erlbaum Associates, Mahwah (1987)
7. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environment - a survey. *IEEE Transactions on Evolutionary Computation* 9(3), 303–317 (2005)
8. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 184–192 (1995)

9. Karaman, A., Uyar, A.S.: A novel change severity detection mechanism for the dynamic 0/1 knapsack problem. In: 10th International Conference on Soft Computing (2004)
10. Karaman, A., Uyar, A.S., Eryigit, G.: The memory indexing evolutionary algorithm for dynamic environments. In: EvoSTOC 2nd European Workshop on Evolutionary Algorithms in Stochastic and Dynamic Environments, pp. 563–573 (2005)
11. Merz, P.: Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation* 12(3), 303–325 (2004)
12. Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Springer, Berlin (2004)
13. Morrison, R.W., DeJong, K.A.: A test problem generator for non-stationary environments. In: Congress on Evolutionary Computation, vol. 3, pp. 2047–2053. IEEE, Los Alamitos (1999)
14. Olsen, A.L.: Penalty functions and the knapsack problem. In: Proceedings of the First IEEE Conference on Evolutionary Computation (1994)
15. Pisinger, D.: Where are the hard knapsack problems? *Computers and Operations Research* 32, 2271–2284 (2005)
16. Rohlfschagen, P., Bullinaria, J.A.: An exonic genetic algorithm with RNA editing inspired repair function for the multiple knapsack problem. In: Proceedings of the 2006 UK Workshop on Computational Intelligence, Leeds, UK, pp. 17–24 (2006)
17. Rohlfschagen, P., Yao, X.: Attributes of combinatorial optimisation. In: Li, X., et al. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 442–451. Springer, Heidelberg (2008)
18. Rothlauf, F.: Representations for genetic and evolutionary algorithms. Springer, Heidelberg (2002)
19. Simões, A., Costa, E.: Using biological inspiration to deal with dynamic environment. In: Proceedings of the Seventh International Conference on Soft Computing (2001)
20. Simões, A., Costa, E.: Using genetic algorithms to deal with dynamic environments: A comparative study of several approaches based on promoting diversity. In: Proceedings of the 2002 Genetic and Evolutionary Computation Conference (2002)
21. Tavares, J., Pereira, F.B., Costa, E.: The role of representation on the multidimensional knapsack problem by means of fitness landscape analysis. In: The 2006 IEEE Congress on Evolutionary Computation (2006)
22. Tinos, R., Yang, S.: Continuous dynamic problem generators for evolutionary algorithms. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pp. 236–243 (2007)
23. Ursem, R.K., Krink, T., Jensen, M.T., Michalewicz, Z.: Analysis and modeling of control tasks in dynamic systems. *IEEE Transactions on Evolutionary Computation* 6(4), 378–389 (2002)
24. van Hemert, J.I., Van Hoyweghen, C., Lukschandl, E., Verbeeck, K.: A "futurist" approach to dynamic environments. In: Branke, J., Bäck, T. (eds.) Proceedings of the Workshop on Evolutionary Algorithms for Dynamic Optimization Problems at the Genetic and Evolutionary Computation Conference, pp. 35–38 (2001)
25. Weicker, K.: Evolutionary algorithms and dynamic optimization problems. Der Andere Verlag (2003)
26. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithms. In: Sarker, R., Reynolds, R., Abbass, H., Tan, K.-C., McKay, R., Essam, D., Gedeon, T. (eds.) Proceedings of the 2003 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2246–2253 (2003)

Impact of Frequency and Severity on Non-Stationary Optimization Problems

Enrique Alba, Gabriel Luque, and Daniel Arias

Departamento de Lenguajes y Ciencias de la Computación

University of Málaga, 29071 Málaga, Spain

{eat,gabriel,darias}@lcc.uma.es

Abstract. Frequency and severity are a priori very influential parameters in the performance of Dynamic Optimization Problems because they establish when and how hard is the change of the target optimized function. We study in a systematic way their influence in the performance of Dynamic Optimization Problems and the possible mathematical correlations between them. Specifically, we have used a steady state Genetic Algorithm, which has been applied to three classic Dynamic Optimization Problems considering a wide range of frequency and severity values. The results show that the severity is the more important parameter influencing the accuracy of the algorithm.

1 Introduction

Many real optimization problems have dynamic nature: job schedule, location of GSM antennae, elevator planning, etc. Evolutionary Algorithms (EAs) have been widely studied as technique for solving Dynamic Optimization Problems (DOPs) [2,3] due to their well-known ability of maintaining a high diversity in the population of tentative solutions.

There are two very important parameters determining the difficulty of DOPs: the frequency and the severity of the changes in the optimized function. Both parameters have been discussed separately in previous works [1,4]. In fact, to the best of our acknowledge, they have never been analyzed at the same time in a scientific approach, i.e., using mathematical metrics, quantitative results, statistical validation, and different problems in a single work. One contribution of the present work is to show how the frequency and the severity are interrelated, establishing under what conditions each factor is more decisive in the performance of a DOP. The degree of severity is measured as the distance between two consecutive optima. In the case of the frequency, we establish a base period of change, being the rest of periods selected in function of the base one.

We have used a Genetic Algorithm (GA) in our study for solving three DOPs with different properties. The tests use a wide range of frequency and severity values. We have also analyzed the impact of the frequency and severity when the GA is combined with a random immigrants strategy [6].

2 DOP Benchmark Used

The DOP benchmark of this work have been selected because they are popular, easily parameterizable in frequency and severity, computationally efficient and they also cover both discrete and continuous domains for a higher impact and utility of our study. Specifically, we have selected three problems [2]: the dynamic knapsack, the moving parabola, and the moving peaks. The goal of the dynamic knapsack problem is to optimally fill a knapsack with objects [1]. The model of change introduced here consists in modifying the maximum capacity of the knapsack. In the moving parabola problem, the base function is a parabolic equation of n dimensions. The dynamic version is defined by shifting the parabola. We have considered in our work linear translations [2]. The moving peaks problem lies in a multidimensional landscape of m peaks, where their positions, heights, and widths can be configured and depend on time. In our model of change only the location of the optimal peak is modified.

3 The Algorithm

In this study we use a steady state Genetic Algorithm (ssGA). We consider that this algorithm is suitable as a baseline one for starting our study. In this algorithm, a single new individual is inserted in the population replacing the worst one. With certain frequency, we introduce a change in the numerical data of the problem, so the fitness of all the individuals must be recalculated. We have considered two different strategies:

- **No Reaction:** the simplest strategy consists in ignoring the change; the population is reevaluated without altering the genetic contents of the individuals. No adaptation to change exists.
- **Random Immigrants** [6]: a percentage of individuals is replaced by random ones following an uniform distribution. This technique introduces additional diversity into the population, and is very useful when the ssGA has converged to a (sub)optimum in the previous problem instance.

Table 1. Metrics used in this work

Accuracy: it measures the relative difference between the best individual in the population and the optimum in that moment	$Acc(t) = \frac{f(best, t)}{optimum(t)}$
Stability: it measures the difference between two consecutive accuracy values.	$Stab(t) = \max(0, Acc(t) - Acc(t - 1))$
Reactivity: this metric measures the ability of the algorithm to react to the changes quickly. It measures the time in which accuracy is similar (depending on the ϵ parameter) to that of time t , and its optimum value is 1	$Reac(t) = \min \{t' - t\} \cup \{maxgen - t\}$ $\left\{ t < t' \leq maxgen, t' \in N, \frac{Acc(t')}{Acc(t)} \geq (1 - \epsilon) \right\}$

4 Methodology of the Study

This section describes the details and procedures for the tests. We use three different metrics [5], then perform a statistical analysis. The metrics are summarized in Table 1. We always perform 100 independent runs of the problem for each experiment, and compute the average of each metric in every. We then perform a Kruskal-Wallis and a multiple comparison tests in order to check the statistical significance. All results were significant. Tables 2 and 3 show the configuration for the ssGA and the problems. We have executed the ssGA in stationary conditions with the aim of determining the average number of evaluations in reaching a global optimum (effort). This value is considered as the base period, being the others selected periods based on it. The different degrees of severity are selected by moving the parabola location (6, 12, 25, 50 or 75 units) or the optimal peak location (1, 5, 10, 20, or 40 units) a certain number of units in each change. In the case of dynamic knapsack, the algorithm oscillates between two different instances. The initial instance has always a weight limit of 104, while the second one is included in {91, 67, 54, 42, 30, 20, 10, 5}. The higher the difference between pairs of weights, the higher the degree of severity.

Table 2. ssGA parameterization

Independent runs	100
Evaluation number	10-period of change
Population size	100
Gen. of the solutions	Uniform
Selection operator	Binary tournament
Replace operator	Worst Replacement
% of random immigrants	0, 10, 30, 50, 70, 100

Table 3. Problems parameterization

	Knapsack	Parabola	Peaks
Codification	Binary	Real	Real
Dimension	17	3	5
Variable range	{0,1}	[-50,50]	[0,100]
Recombination	SPX	SBX ($p_c = 0, 7$)	($p_c = 0, 7$) ($\eta_c=20$)
Mutation	Bit Flip ($p_m = 1/17$)	Polynomial ($p_m = 0, 3$)	($\eta_m=20$)
Base period (b)	700	1000	1500
Periods	0.5b 1b 1.5b 2b 4b	0.25b 0.5b 1b 1.5b 2b	0.3b 0.6b 1b 1.3b 1.6b 2b
Severity criterion	Hamming dist.	Euclidean dist.	
Others	$\lambda = 20$	-	10 Peaks

5 Experimental Analysis

We analyze in the first part of this section how severity and frequency affect the accuracy. In the second, reactivity and stability will be analyzed.

5.1 Impact of the Frequency and the Severity on the Accuracy

Fig. 1 shows a representative example of the mean accuracy as a function of the severity and the period of change. On the one hand, we can observe that the higher the period of change (low frequency) the higher the accuracy. This is due to the ssGA computes more evaluations, so it can adapt itself better to the change, and find more accurate solutions. On the other hand, the impact on the accuracy is negative when the severity is increased. The distance between the consecutive optima increases according to the degree of severity, so the GA needs more effort for adapting to the new instance and the accuracy decreases. With the aim of analyzing quantitatively the effect of the frequency and the severity on the accuracy, we have modeled the experimental results by means of

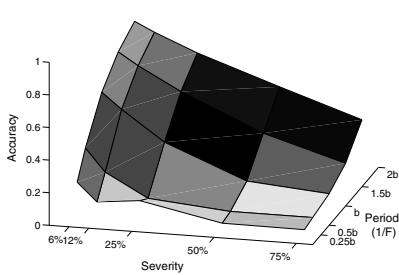


Fig. 1. Accuracy depending on the frequency and the severity in the parabola problem

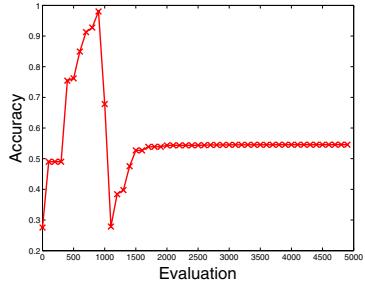


Fig. 2. Evolution of the accuracy in a moving peaks execution. The optimal peak changes its location every 1000 evaluations.

mathematical linear regression because its simplicity, its accuracy (its error is lower than 2%), and it eases the interpretation of the results. Specifically, the equations present the form:

$$\text{Accuracy} = w_i + w_s \cdot S + w_f \cdot F, \quad (1)$$

where S is the severity, F the frequency (normalized between 0 and 1), and the w_s and w_f coefficients show the influence of F and S on the accuracy (w_s and w_f lower than 0 involve a negative impact on the accuracy). The parameter w_i is an *approximation* of the accuracy when the static problem is solved because we have not worked under stationary conditions. The resulting equations are:

$$\text{Accuracy}_{\text{Knapsack}} = 1.15 - 0.54 \cdot S - 0.15 \cdot F \quad (2)$$

$$\text{Accuracy}_{\text{Peaks}} = 0.96 - 1.00 \cdot S - 0.06 \cdot F \quad (3)$$

$$\text{Accuracy}_{\text{Parabola}} = 0.71 - 0.59 \cdot S - 0.47 \cdot F \quad (4)$$

It is evident that both parameters impact negatively on the accuracy. As we said before, this is an expected result because the increment of the severity and/or the frequency makes more complex the dynamic problem. Moreover, the severity is more decisive than the frequency ($w_s > w_f$).

Frequency and severity can play different roles in the performance depending on the problem. In the parabola problem, the frequency is more important than in the others. The search space of this problem is unimodal, so the low frequencies are the key for reaching the global optima (or very accurate solutions). However, the situation changes when we deal with multimodal problems (i.e., knapsack and peaks). In these cases, w_f is very low, even insignificant with respect to w_s , being the severity the most influential parameter. To illustrate this fact, Fig. 2 shows that after a big change (at 1,500 evaluations approximately) the accuracy keeps constant because the ssGA is trapped in a local minimum (close to the previous global optimum). The main conclusion of this part is that the severity is the most influential parameter in the accuracy of the algorithm.

Table 4. Equation coefficients for the accuracy according to the different replacement percentages

% Replace	Knapsack			Peaks			Parabola		
	w_i	w_s	w_f	w_i	w_s	w_f	w_i	w_s	w_f
0	1.15	-0.54	-0.15	0.96	-1.00	-0.06	0.71	-0.59	-0.47
10	1.12	-0.39	-0.15	0.97	-0.94	-0.08	0.75	-0.23	-0.66
30	1.09	-0.29	-0.14	0.95	-0.89	-0.07	0.77	-0.14	-0.71
50	1.07	-0.24	-0.13	0.97	-0.88	-0.10	0.78	-0.10	-0.72
70	1.06	-0.21	-0.12	0.97	-0.86	-0.09	0.78	-0.07	-0.72
100	1.05	-0.16	-0.12	0.94	-0.33	-0.31	0.79	-0.02	-0.75

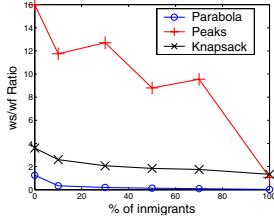


Fig. 3. Ratio between the severity and frequency coefficients (w_s/w_f) according to the replacement percentage

Our next step consists in analyzing the effects of the severity and the frequency when the Random Immigrants [6] strategy is added to the ssGA. With the aim of covering a wide range of percentage values (including the extreme ones), we have studied five different replacement percentages: 10%, 30%, 50%, 70% and 100%. Table 4 shows the different coefficients obtained from applying linear regression to the experimental results. The w_i and w_f coefficients do not change (lower than 0.1 unit) in the multimodal problems (peaks and knapsack), while the w_f changes more drastically in the parabola problem (as we said previously, in unimodal problems the frequency plays an important role). If we analyze the ratio (w_s/w_f), we can observe how this value decreases as the replacement percentage increases (Fig. 3). The reason is the additional diversity added to the population by the random immigrants which is specially useful for drastic changes or when the population has converged to a local optimum. The random immigrants strategy allows to reduce the negative effect of a large severity in the accuracy of the algorithm. This is an expected result: as we increase the replacement percentage, the population is closer a restarted one, so the dynamic problem is solved as a succession of static ones.

5.2 Stability and Reactivity

This subsection analyzes the influence of the frequency and the severity in the other two metrics proposed by Weicker [5]: the stability and the reactivity. The expressions obtained from the experimental results are shown in Table 5.

Table 5. Equation coefficients for the stability and the reactivity

Metric	Knapsack			Peaks			Parabola		
	w_i	w_s	w_f	w_i	w_s	w_f	w_i	w_s	w_f
Stability	-0.030	0.140	0.050	0.006	-0.009	0.020	0.080	-0.070	0.001
Reactivity	-0.080	1.020	0.790	1.630	-0.005	-0.550	1.830	0.830	-0.690

For these metrics, the results are dependent of the problem. The reason are the special features of the different search spaces. Fig. 4 illustrates this fact.

We want to show traces of two opposite scenarios for each problem. In the left picture, a drastic change and a high frequency is considered. The oscillation

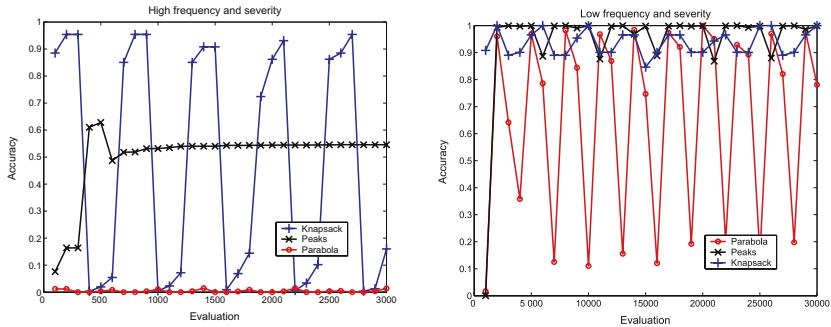


Fig. 4. Evolution of the accuracy in the studied problems for opposite configurations in frequency and severity

(characteristic which determines the stability and the reactivity values) of the accuracy is different in each problem. Thus, in the dynamic knapsack, high penalizations are applied to the solutions each time the weight limit is reduced, so the accuracy suffers high oscillations. In the parabola problem the oscillation is very smooth because the algorithms have no time to achieve accurate solutions (the accuracy is always low), while in the moving peaks problem the accuracy is constant when the algorithm has converged to a local minimal (see Fig. 2).

The right graph of Fig. 4 represents a low severity and frequency. In this case, the behaviour in the oscillation of the accuracy is also dependent of the problem: the evolution in the knapsack trace is smoother than before (the weight limits between changes are very similar being the penalizations very low). In the other two problems, the oscillation is higher than in the other case (above all in the moving parabola) because of the continuous adaptation to the new optima.

6 Conclusions

This work studies the influence of two important parameters in non-stationary environments: the severity and the frequency between two adjacent optimized functions. We have used a ssGA for solving three different DOPs: the dynamic knapsack, the moving parabola, and the moving peaks. The experimental tests have covered a wide range of frequency and severity values.

The experimental results have been analytically expressed into equations whose free variables are the two studied parameters. We do not observe a direct relation of these parameters in the stability and the reactivity metrics. However, in the accuracy metric (the most important one), we can obtain some conclusions. In general, the severity is the most influential parameter; its effect is more important in multimodal problems. In unimodal problems (e.g., moving parabola), low frequencies help to reach accurate solutions.

We have also studied the effects of adding the random immigrants strategy to the ssGA. The results of this part show how the increment of the replacement percentage decreases the negative effects of high severities thanks to the additional diversity introduced by the technique.

The study of the influence of frequency and severity using other algorithms and strategies are possible lines of further work. This way, we can check if the drawn conclusions of this work can be extended in other contexts.

Acknowledgments

This work has been partially financed by the projects P07-TIC-03044 (DIRI-COM) and TIN2008-06491-C04-01 (M*).

References

1. Alba, E., Saucedo, J.F., Luque, G.: MIC 2005 Post Conference Volume on Metaheuristics - Progress in Complex Systems Optimization, ch. 13. A Study of Canonical GAs for NSOPs, pp. 245–260. Springer, Heidelberg (2007)
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Dordrecht (2002)
3. Morrison, R.W.: Designing EAs For Dynamic Environments. Springer, Heidelberg (2004)
4. Weicker, K.: An analysis of dynamic severity and population size. In: Deb, K., et al. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 159–168. Springer, Heidelberg (2000)
5. Weicker, K.: Performance Measures for Dynamic Environments. LNCS, pp. 64–76. Springer, Heidelberg (2003)
6. Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Proc. of GECCO, pp. 1115–1122. ACM, New York (2005)

A Critical Look at Dynamic Multi-dimensional Knapsack Problem Generation

Şima Uyar and H. Turgut Uyar

Istanbul Technical University
`{etaner,uyar}@itu.edu.tr`

Abstract. The dynamic, multi-dimensional knapsack problem is an important benchmark for evaluating the performance of evolutionary algorithms in changing environments, especially because it has many real-world applications. In order to analyze the performance of an evolutionary algorithm according to this benchmark, one needs to be able to change the current problem in a controlled manner. Several methods have been proposed to achieve this goal. In this paper, we briefly outline the proposed methods, discuss their shortcomings and propose a new method that can generate changes for a given severity level more reliably. We then present the experimental setup and results for the new method and compare it with existing methods. The current results are promising and promote further study.

Keywords: Dynamic environments, dynamic problem generators, constrained problems, change severity, evolutionary algorithms.

1 Introduction

The multi-dimensional knapsack problem (MKP) [1] is one of the most commonly used constrained optimization problems for analyzing the performance of evolutionary algorithms (EAs) because it has a wide range of real world applications such as cargo loading and budget management. The dynamic version of the MKP (dMKP) is also commonly used in changing environments.

To compare different EA approaches on dynamic problems, changes have to be performed in a controlled manner. For example, if the approaches are being compared based on their performance under changes of different severity levels, then the dynamic environment generator has to be able to generate change instances of controlled severity. With the presence of constraints, implementing changes with controlled severities becomes more difficult.

In this study, we first analyze existing dMKP generation methods in literature and point out their shortcomings. Then we discuss the properties a good generator should possess and propose metrics for comparisons. Based on our observations, we propose a new approach to create dMKP instances with controlled severities. We then perform experiments using these metrics on sample MKP instances. This is a preliminary step towards analyzing the nature of the severity of changes in the dMKP and designing a good dMKP instance generator.

2 The Dynamic Multi-dimensional Knapsack Problem

Given a set of resources with different capacities and a set of items with different profits and resource consumptions, the MKP can be defined as maximizing the total profit of selected items while adhering to the capacity constraints of the resources. In dMKP, the problem instance changes over time. Currently there are two main approaches for generating dMKP instances:

Changing the problem parameters. In the *dMKP Generator* proposed in [2], the capacities, profits and resource consumptions are multiplied by a normally distributed random variable. The standard deviation of the random variable determines the severity of the change. This generator was used to test EA performances or analyze their behavior, e.g. in [3].

Changing the mapping of the points in the search space. In the *XORing generator* proposed in [4], dynamic test problems can be generated from any binary encoded stationary problem. For each change, a bit string mask is generated which is applied to each solution using an exclusive-or operation before the fitness value is calculated. Changing the bits in the mask changes the position of the points in the search space and the number of changed bits in the mask determines the severity of the change. This generator was used in several studies, e.g. in [5].

We observe that the proposed approaches have significant shortcomings:

Generators which change the profits, resource consumptions and capacities, like the dMKP generator, do not take the direction of the change into consideration. However, a change which tightens the constraints might be more severe than a change of the same magnitude that relaxes the same constraints. Besides, in the dMKP generator, since all of these values are multiplied by a normally distributed random number around 0, changes in the different components may have contradictory effects.

Generators that only change the solution mapping but not the problem instance, like the XORing generator, do not take into account the presence of constraints. They define severity based on the number of changed bits in the mapping. However, this definition might not apply to all cases. For example, assume that there is one resource with capacity 92 and six items with profits 100, 50, 35, 5, 3, 1 with the resource consumption value for each item being equal to its profit. When the XOR mask is $< 000000 >$, the best solution for this problem is $< 011101 >$ with a fitness value of 91. Changing the mask to $< 100000 >$ toggles the inclusion of the first item with resource consumption value 100 and causes all feasible individuals in the population to become infeasible. However, changing the mask to $< 000001 >$ has the same severity under this definition, whereas it toggles the inclusion of the last item with resource consumption value 1, leaving all feasible individuals in the population still feasible.

These observations motivated us to question whether severity should be measured based on the actual amount of change in the problem instance or based on how the search algorithm sees this change. We believe that changes should be performed based on the problem instance itself and severity should be defined

based on how much the problem instance changes. Changing the mapping implies that there is no difference in including or removing any one of the items in the optimal solution. This assumption ignores the properties of the underlying problem and reduces it to searching for one or more bit patterns. In this case, the changes do not correspond to what actually happens in the real problem instance. However, the main purpose for using the dMKP as a benchmark is to show how the algorithms perform on a real-world problem.

To measure the true severity of the change in the dMKP, we propose to look at several metrics before and after the change:

1. The ratio of feasible and infeasible solutions in the search space: It was shown for the dynamic single knapsack problem that the amount and direction of the change in this ratio is related to the amount and direction of the change in the constraint [6]. Based on the landscape properties of the penalty-based MKP solution approach in [7] a similar observation holds for the dMKP.
2. The ratio of feasible and infeasible individuals in the EA population: Based on this ratio, the absolute value of the amount of change in the problem instance can be estimated. Also the direction of the change in this ratio may indicate the direction of the change in the problem instance.
3. The change ratio in the pairwise fitness comparisons of the individuals in the population: An individual which is better than another may become worse after a change. In an EA, one of the driving forces of evolution is selection. Especially in selection methods that rely on the pairwise "better fitness" relationship of the individuals, if these relationships are reversed, selection cannot be effective in guiding the EA towards better solutions.

Based on the above observations and discussions, we propose a dMKP instance generator which takes into account the severity and the direction of the change in the actual problem instance. For simplicity, we only change the capacity constraints as given in the equation below. After determining the total amount of desired capacity change and direction, this amount is distributed between the resources approximately equally.

$$c_i \leftarrow c_i + N(\mu, \sigma) \quad \mu = \pm \frac{C_T * \rho}{R}$$

where c_i is the capacity of item i , μ and σ are the average and standard deviations of the normal distribution respectively. The average is calculated also as given in the equation where C_T is the total capacity of all the resources, ρ is the change severity and R is the total number of resources.

3 Experiments

The aim of the experiments is to show that the metrics explained in Section 2 can be used to measure the change severity in the dMKP and to compare the different dMKP instance generators using these metrics. The following MKP instances from the OR-Library have been used in the experiments: flei (10/20),

hp1 (4/28), pb1 (4/27), pb4 (2/29), pb5 (10/20), pet2 (10/10), pet3 (10/15), pet4 (10/20), pet5 (10/28), weing3 (2/28), weish03 (5/30) and weish28 (5/90).¹ The numbers in parenthesis show the sack and item counts for each instance.

In the following sections, the dMKP generator will be denoted as M1, the XORing generator as M2 and the proposed generator as M3- and M3+ when the capacities are decreased and increased respectively. For all experiments, two severity levels of 0.1 and 0.25 are tested.

For the search space metric experiments, the problems with relatively large sizes (pb4, pet5, weing3, weish03 and weish28) have been used. 20 random samplings with $\min(1000000, 2^l/100)$ points each are performed (l is the item count) for each case. And for each sampling 20 random changes are implemented. Results are reported over 400 values for each problem instance and method. For the smaller sized problems, the entire search space has been enumerated. Since the findings were very similar with the sampled experiments, the results for the smaller sized problems will not be included.

Since the purpose of this study is not designing good EAs for the dMKP, a simple generational EA with no elitism is used in the EA experiments. We used the binary representation where each chromosome is a bit string with each bit showing whether an item is included in the solution. This representation divides the search space into feasible and infeasible regions. Penalty methods are commonly used to handle infeasible solutions, such as the method proposed and shown to be good in [7]. This method applies very large penalty values to infeasible solutions, thus guiding the search towards feasible regions quickly. Selection is done through binary tournaments. Uniform crossover with bit-flip mutation is used in the reproduction phase. The chromosome length is equal to the item count, population size is 100, uniform crossover rate is 1.0, mutation rate is $1/l$ where l is the chromosome length.

For each run of the EA, only one random change is applied. For each problem, method and severity combination, the EA is run 100 times. Initially, each problem instance is run without any changes and a medium change period where the population has not yet fully converged is approximately determined. This period is taken as 300, 1000, 2500, 3000, 4000, 6000 and 25000 iterations for problem instances with item counts of 10, 15, 20, 27, 28, 30, 90 respectively. After the change, the EA is allowed to run for a $\min(300, sp/10)$ of iterations until it is terminated (sp is the number of iterations before the change) and, in order to measure the performance of the EA, the first iteration in which the best solution is found (first hit time - FHT) and the first iteration in which a feasible individual is found (first feasible time - FFT) are noted.

The results of the experiments will be reported through tables. In each table the first value in each cell shows the mean and the second value shows the standard error of the mean.

The difference between the ratios of feasible solutions after and before the change in the sampled search space are calculated for each problem instance, method and severity level. Due to space limitations, only the averages are given

¹ <http://www.brunel.ac.uk/depts/ma/research/jeb/orlib/mknapsackinfo.html>

Table 1. Summary of feasible ratio changes in the search space

	M1	M3-	M3+
0.1	-0.063 , 0.014	-0.144 , 0.028	0.120 , 0.019
0.25	-0.231 , 0.044	-0.340 , 0.057	0.253 , 0.033

in Table 1. Since M2 applies change only to the mapping of the solutions, the actual landscape is not changed. All feasible ratios remain constant, therefore M2 results are not reported.

M1 has a tendency to decrease the feasible ratio in the search space because it changes the values for both the resource consumptions of the items and the resource capacities. For each MKP instance, the number of items is higher than the number of resources. An increase in the total resource consumptions will most likely be less than the total increase in the capacities.

For both M3- and M3+, the change in the feasible ratios reflects the direction and the magnitude of the change as intended. Even though for M1, the change in the ratios of the feasibles does not directly reflect the actual change severity, it increases as the severity of the change as applied by the changer increases.

If we look at the individual results for each method and severity for each problem separately (omitted here due to lack of space), we can see that the standard errors for M1 are very high. This is because changes in different components interact with each other.

As a result, this metric is irrelevant for M2 and inapplicable for M1 because of the high standard error. It can be used reliably with M3.

The change in the ratio of feasible individuals and the ratio of reversed fitness comparison relationships for the EA experiments are given in Table 2.

Table 2. Summary of EA experiment results

a. Feasible ratio changes	b. Pairwise fitness relationship changes			
	M1	M2	M3-	M3+
0.1	-0.148	-0.157	-0.322	0.131
	0.026	0.046	0.049	0.015
0.25	-0.389	-0.285	-0.613	0.194
	0.05	0.06	0.04	0.02

	M1	M2	M3-	M3+
0.1	0.232	0.307	0.291	0.179
	0.023	0.013	0.037	0.019
0.25	0.390	0.435	0.418	0.281
	0.03	0.01	0.04	0.02

M1 and M2 tend to cause changes which decrease the ratio of feasibles. The direction of the changes for M3- and M3+ are correct but the change in the ratio of feasibles for M3- is higher than that of M3+. M3+ causes the smallest changes in both metrics. However even though M2 causes the largest changes in the fitness switches, that is not the case for the change in the feasible ratios. Therefore, these two metrics provide complementary information regarding the magnitude and direction of the change and should be used together.

The results of the EA performance tests based on the FFT and FHT have been omitted due to space limitations, but they can be summarized as follows: For M2, if there is a change in the feasibles ratio in the population, this has

a correlated effect on the FFT and FHT results. The FFT for M3- is higher than M3+ but the FHT values for M3- are better. This is due to the penalty approach [7] used. In the MKP, the optimum solutions lie on the boundary between the feasible and infeasible regions. The very large penalty values drive the population quickly to the boundary. When the capacities are decreased, a large portion of the population becomes infeasible. The population moves quickly to the new boundary. When the capacities are increased, all feasible individuals remain feasible. Their FFT is always close to 1 but it takes them longer to get to the boundary, hence the larger FHT values. This shows us that looking at the FHT value is better if we are concerned with the offline behavior of the EA but the FFT becomes more important in assessing the online performance.

4 Conclusions and Future Work

Our findings show that in dMKP, changes should be performed based on the problem instance and severity should be defined based on how much the problem instance changes. We also show that the approach proposed in this study creates more realistic dMKP instances for which the severity can be better controlled. The direction of the change can be manipulated and different severity levels corresponding to real-world changes can be generated more reliably.

This is a preliminary step towards analyzing the nature of the severity of changes in dMKP and designing a good dMKP instance generator. The current results are promising and promote further study. The parameter settings for the proposed approach should be further explored to enhance precision. Also the metrics proposed in [2] will be considered and evaluated for our purpose.

References

1. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
2. Branke, J., Salihoglu, E., Uyar, S.: Towards an Analysis of Dynamic Environments. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 1433–1439. ACM, New York (2005)
3. Branke, J., Orbayi, M., Uyar, S.: The role of representations in dynamic knapsack problems. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 764–775. Springer, Heidelberg (2006)
4. Yang, S., Yao, X.: Experimental Study on Population-based Incremental Learning Algorithms for Dynamic Optimization Problems. *Soft Computing* 9(11), 815–834 (2005)
5. Yang, S., Yao, X.: Population-Based Incremental Learning with Associative Memory for Dynamic Environments. *IEEE Trans. on Evolutionary Comp.* (2008)
6. Karaman, A., Uyar, A.S., Eryigit, G.: The Memory Indexing Evolutionary Algorithm for Dynamic Environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 563–573. Springer, Heidelberg (2005)
7. Gottlieb, J.: Evolutionary Algorithms for Combinatorial Optimization Problems, Phd Thesis, Technical University Clausthal, Germany (1999)

Evolutionary Freight Transportation Planning

Thomas Weise¹, Alexander Podlich², Kai Reinhard², Christian Gorlitz³,
and Kurt Geihs¹

¹ Distributed Systems Group, University of Kassel, 34121 Kassel, Germany

² Micromata GmbH Kassel, Marie-Calm-Strae 3, 34131 Kassel, Germany

³ BIBA – Bremer Institut für Produktion und Logistik GmbH

Abstract. In this paper, we present the freight transportation planning component of the *INWEST* project. This system utilizes an evolutionary algorithm with intelligent search operations in order to achieve a high utilization of resources and a minimization of the distance travelled by freight carriers in real-world scenarios. We test our planner rigorously with real-world data and obtain substantial improvements when compared to the original freight plans. Additionally, different settings for the evolutionary algorithm are studied with further experiments and their utility is verified with statistical tests.

1 Introduction

With the steadily increasing freight traffic resulting from trade inside the European Union and global import and export as well [1], the need for intelligent solutions for the strategic planning of logistics is growing steadily. Such a planning process has the goal of

1. increasing the profit by
2. ensuring the on-time collection and delivery of all parcels,
3. utilizing all available means of transportation (rail, trucks) efficiently, i. e., reducing the total transportation distances by using the capacity of the vehicles to the full, while
4. reducing the CO₂ production in order to become more environment-friendly.

Obviously, the last point is a side-effect of the others. By reducing the total distance covered and by transporting a larger fraction of the freight by (inexpensive) trains, not only the hours of work of the drivers and the costs are reduced, but also the CO₂ production.

Additionally, an efficient freight planning process has two other constraints. First, it must be dynamic and able to react to traffic jams by re-routing freight vehicles. Second, experience has shown that hiring external carriers for a small fraction of the freight in local areas can often reduce the number of required own tours significantly and thus, increase the relative utilization of the own means of transportation (see, for instance, Fig. 3b). Therefore, freight planning is a multi-objective optimization problem and the human operators must also be provided with partial solutions to select from.

In this paper, we present the freight transportation planning component of *INWEST*, a joint project funded by the German Federal Ministry of Economics and Technology, which fulfills all these requirements. In the following section, we discuss the general requirements of the logistics departments of the project partners which specify the framework for our freight planning component.¹ These specific conditions ruled the related approaches outlined in Section 3 infeasible. In Section 4, we present freight planning as a multi-objective evolutionary optimization [2] problem. The problem-specific representation of the solution candidates and the intelligent search operators working on them are introduced, as well as the objective functions derived from the requirements already stated in this section. Our approach has been tested in many different scenarios and the experimental results are summarized in Section 5. The paper concludes with a discussion of the results and future work in Section 6.

2 Model Based on the Real-World Situation

The basic unit of freight considered in this work is a swap body b , a standardized container (C 745, EN 284 [3]) with an extent of roughly $7.5\text{m} \times 2.6\text{m} \times 2.7\text{m}$ and special appliances for easy exchange between transportation vehicles or railway carriages. Logistics companies like the *DHL* own thousands of such swap bodies and we refer to their union as set B .

The set of all possible means of transportation will be referred to as F in the following. All trucks $tr \in F$ can carry $\hat{v}(tr) = 2$ such swap bodies at once whereas the capacity limits of trains $z \in F$ are usually somewhere between 30 and 60 ($\hat{v}(z) \in [30..60]$). Trains have fixed routes, departure, and arrival times. Freight trucks can move freely on the map, but must usually perform cyclic tours, i.e., return to their point of departure by the end of the day, so that the human drivers are able to return home.

The clients and the depots of the logistics companies together form roughly 1000 locations from which freight may be collected or to which it may be delivered. We will refer to the set of all these locations as L . Each transportation order has a fixed time window $[\check{t}_s, \hat{t}_s]$ in which it must be collected from its source $l_s \in L$ and a destination location and time window $[\check{t}_d, \hat{t}_d]$ in which it must be delivered to its destination $l_d \in L$. It furthermore has a volume v which is an integer multiple of the capacity of a swap body. Hence, a transportation order o can fully be described by the tuple $o = \langle l_s, l_d, [\check{t}_s, \hat{t}_s], [\check{t}_d, \hat{t}_d], v \rangle$. Depending on the day of week and national holidays etc., between 100 and 3000 such orders have to be processed per day. In the following, all orders which require more than one ($v > 1$) swap body will be split up into multiple orders requiring one swap body ($v = 1$) each.

The result of the planning process is a set R of tours. Each single tour r is described by a tuple $r = \langle l_s, l_d, f, \check{t}, \hat{t}, b, \underline{o} \rangle$ where l_s and l_d are the start and

¹ That partners in the project *INWEST* (Intelligente Wechselbrücksteuerung, funded by the German Federation) are the *Deutsche Post AG*, *DHL*, the *Micromata GmbH*, *BIBA*, and *OHB Teledata GmbH*; see <http://www.inwest.org/> [accessed 2008-10-29].

destination locations, \check{t} and \hat{t} are the departure and arrival time, $\underline{b} = \{b_1, b_2, \dots\}$ is a set of swap bodies which are carried by the vehicle $f \in F$ and contain the goods assigned to the orders $\underline{o} = \{o_1, o_2, \dots\}$.

Additional to the constraints mentioned in the introduction, a freight planning system must ensure that the tours computed are *physically sound*. Neither freight vehicles nor orders or swap bodies can be part of more than one tour at a time and the capacity limits of all involved means of transportation must be respected. If some of the freight is carried by trains, the fixed halting locations of the trains as well as their assigned departure and arrival times must be considered. Of same importance are the restrictions imposed on the runtime of the optimization process which must not exceed one day.

3 Related Work

The approaches discussed in the literature on freight transportation planning can roughly be divided into two basic families: deterministic and stochastic or metaheuristic methods. Especially the metaheuristic optimization methods received more and more attention during the past years. The quality of solutions produced by them is often much higher than that obtained by classical heuristics.

Well-known approaches for different types of vehicle routing problems and freight transportation planning are Tabu Search [4–7], Simulated Annealing[8, 9], ant systems [10, 11], and especially evolutionary algorithms [12–15]. Most of these publications outline special types of vehicle routing problems (e.g., the vehicle routing with time windows or vehicle routing with back hauls). Usually, these problems are solved with single-objective optimization enriched with problem-specific constraints. The size of such problems is typically roughly around 100 customers or orders, as is the case in most of the example data sets in [16, 17].

The authors of the publications mentioned in the text above often indicate that metaheuristics succeed only when a good deal of domain knowledge is incorporated. This holds not only for vehicle routing, but is the case in virtually every application of global optimization [18, 19]. Nevertheless, such knowledge is generally used as an extension, as a method to tweak generic operators and methods. In this work, we have placed problem-specific knowledge in the center of the approach.

4 Evolutionary Approach

Evolutionary algorithms are a family of nature-inspired optimization algorithms which utilize natural processes such as selection and reproduction in order to refine a set (population) of solution candidates iteratively [20, 21]. This cycle starts with the evaluation of the objective values of the solution candidates. Based on these results, a relative fitness is assigned to each solution candidate in the population. These fitness values are the criteria on which selection algorithms operate that pick the most promising individuals for further investigation while discarding the lesser successful ones. The solution candidates which managed to enter the so-called *mating pool* are then reproduced, i. e., combined via crossover

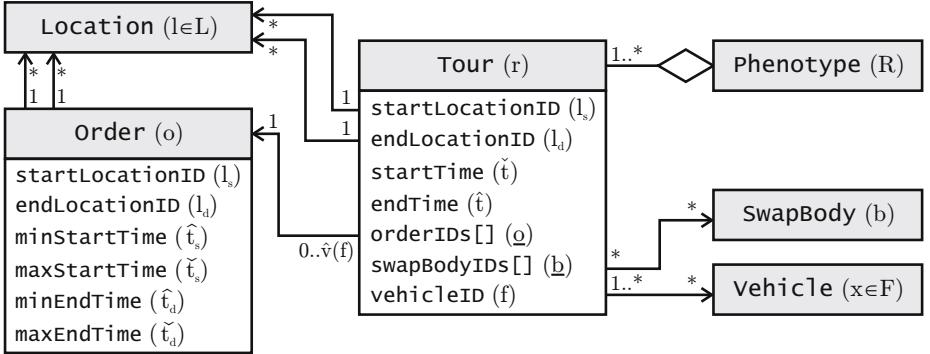


Fig. 1. The structure of the phenotypes R

or slightly changed by mutation operations. After this is done, the cycle starts again in the next generation.

4.1 Search Space

When analyzing the problem structure outlined in Section 2, it becomes very obvious that standard encodings such as binary [22] or integer strings, matrixes, or real vectors cannot be used in the context of this special logistics planning task. Although it might be possible to create a genotype-phenotype mapping capable of translating an integer string into a tuple r representing a valid tour, trying to encode a set R of a variable number of such tours in an integer string is not feasible. First, there are many substructures of variable length such as the sets of orders \underline{o} and swap bodies \underline{b} involved in a tour. Also, it would practically be impossible to ensure that the required *physical soundness* of the tours given that the reproduction operations would randomly modify the integer strings.

In our work, we adhered to the premise that *all solution candidates must represent correct, i. e., physically sound, solutions and none of the search operations is allowed to violate this correctness*. A solution candidate R not necessarily contains a complete plan which manages to deliver every order. Instead, partial solutions (as demanded in Section 1) are admitted, too. However, all individuals in the populations not only respect the laws of physics but also requirements such as cyclic routes for trucks.

In order to achieve such a behavior, it is clear that all reproduction operations of our evolutionary algorithm must have access to the complete tuples r . Therefore, the phenotypes are not encoded at all but instead, the plan objects in their native representations as illustrated in Figure 1.

4.2 Search Operations

By using this explicit representation, the search operations have full access to all information in the freight plans. Standard crossover and mutation operators are, however, now longer applicable. Instead, intelligent operators have to be

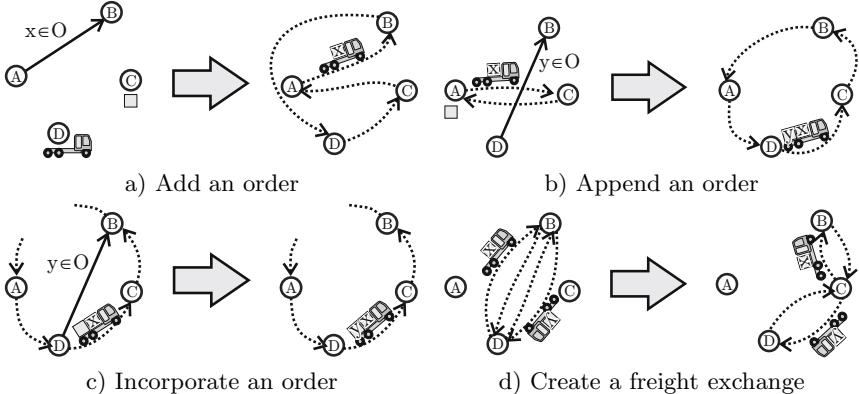


Fig. 2. Some mutation operators from the freight planning EA

introduced which respect the correctness of the solution candidates. In total, three crossover and sixteen mutation operations have been defined, each dealing with a specific constellation in the phenotypes and performing one distinct type of modification. During the evolution, individuals to be mutated are processed by a randomly picked operator. If the operator is not applicable because the individual does not exhibit the corresponding constellation, another operator is tried and so on. Two individuals to be combined with crossover are processed by a randomly selected operator as well.

Obviously, we cannot give detailed specifications on all twenty genetic operations [23] (including the initial individual creation) in this paper. Instead, we will outline the four mutation operators sketched in Figure 2 exemplarily.

The first operator (Fig. 2a) is applicable if there at least one order $x \in O$ is not delivered if the plan in the input phenotype R was carried out. This operator chooses randomly from all available means of transportation. *Available* in this context means not involved in another tour for the time between the start and end time of the order. The freight transporters closer to the source of the order are picked with higher probability. Then, a swap body is allocated in the same manner. This process leads to one to three new tours which are added to the phenotype. If the transportation vehicle is a truck, a fourth tour is added which travels back to the origin.

Fig. 2b illustrates one operator which extends a given set of tours by including a new order in the route of a vehicle. The mutator sketched in Fig. 2c does the same if an additional order can be included in already existing tours. For all operators which add new orders, swap bodies, or tours to the solution candidates, similar operations which remove these elements are provided.

Especially interesting is the “truck-meets-truck” mechanism. Often, two trucks are carrying out deliveries in opposite directions ($B \rightarrow D$ and $D \rightarrow B$ in Fig. 2d). If the time windows of the orders allow it, the two involved trucks can meet at a halting point somewhere in the near of their routes and exchange their freight. This way, the total distance that they have to drive can almost be halved.

The crossover operators combine two phenotypes by intermixing their tours. In this process, tours which belong together such as those created in the first mutator are kept together.

4.3 Objective Functions

The freight transportation planning process run by the evolutionary algorithm is driven by three objective functions. These functions, all subject to minimization, are based on the requirements stated in Section 1 and are combined via Pareto comparisons [2, 20] in the subsequent fitness assignment processes.

- f₁*: **Order Completion.** One of the most important aspects of freight planning is to deliver as many of the orders as possible. Human operators would need to hire external carriers for orders which cannot be delivered (due to insufficient resources, for instance). Therefore, the first objective function returns the number of orders not considered in a freight plan.
- f₂*: **Kilometers Driven.** By using a distance matrix kept in memory, the second objective function determines the total distance covered by all vehicles involved. Minimizing this number will lead to less fuel consumption and thus, lower costs and lesser CO₂ production.
- f₃*: **Full Utilization of the Capacities.** The third objective function minimizes the spare capacities of the vehicles involved in tours. In other words, it counts the total volume left empty in the swap bodies on the road and the unused swap body slots of the trucks and trains.

5 Experiments

We have evaluated our freight planning system rigorously. Therefore, we have carried out a series of tests according to the full factorial design of experiments paradigm [24, 25]. These experiments (which we will discuss in Section 5.1) are based on a single real-world set of orders. The results of additional experiments performed with different datasets are outlined in Section 5.2. All data used has been reconstructed from the actual order database of the project partner *DHL*, one of the largest logistics companies worldwide.

The experiments were carried out using a simplified distance matrix for both, the EA and the original plans. Furthermore, legal aspects like statutory idle periods of the truck drivers have not been incorporated. However, an analysis of the results showed that these were not violated by the plans. The EA is able to utilize trains, but since the original plans did not do this, we turned off this feature in the experiments too, in order to keep the results comparable.

5.1 Full Factorial Tests

The full factorial test series is based on a set of 183 orders reconstructed from one day in December 2007. The original freight plan for these orders contained 159 tours which covered a total distance of $d = 19\,109$ km. The capacity of the vehicles involved was used to 65.5%.

Table 1. The top-ten evaluation results

#	mr	cr	cp	el	ps	ss	fa	ar	at	gr	gt	et	$e\tau$	d
1.	0.8	0.4	1	1	1000	1	1	$^{10/10}$	341	$^{10/10}$	609	3078	3 078 500	15 883 km
2.	0.6	0.2	1	0	1000	1	1	$^{10/10}$	502	$^{10/10}$	770	5746	5 746 500	15 908 km
3.	0.8	0.2	1	1	1000	1	1	$^{10/10}$	360	$^{10/10}$	626	4831	4 831 000	15 929 km
4.	0.6	0.4	1	0	1000	1	1	$^{10/10}$	468	$^{10/10}$	736	5934	5 934 000	15 970 km
5.	0.6	0.2	1	1	1000	1	1	$^{10/10}$	429	$^{10/10}$	713	6236	6 236 500	15 971 km
6.	0.8	0.2	1	0	1000	1	1	$^{10/10}$	375	$^{10/10}$	674	5466	5 466 000	16 003 km
7.	0.8	0.4	1	1	1000	1	0	$^{10/10}$	370	$^{10/10}$	610	5691	5 691 500	16 008 km
8.	0.8	0.2	1	0	1000	0	1	$^{10/10}$	222	$^{10/10}$	450	6186	6 186 500	16 018 km
9.	0.8	0.4	0	0	1000	0	1	$^{10/10}$	220	$^{10/10}$	463	4880	4 880 000	16 060 km
10.	0.8	0.2	0	1	1000	0	0	$^{10/10}$	277	$^{10/10}$	506	2862	2 862 500	16 071 km

- ss.** The parent individuals in the population are either discarded (generational, $ss = 0$) or compete with their offspring (steady-state, $ss = 1$).
- el.** The best solution candidates were either preserved (elitism, $el = 1$) or not (no elitism, $el = 0$).
- ps.** Three different population sizes were tested: $ps \in \{500, 1000, 2000\}$
- fa.** Either simple Pareto-Ranking [2] ($fa = 0$) or an extended assignment process with sharing ($fa = 1$, called *variety preserving* in [20]) were applied.
- cp.** The simple convergence prevention (SCP) method proposed in [20] was either used ($cp = 1$) or not ($cp = 0$).
- mr/cr.** Different settings for the mutation rate $mr \in \{0.6, 0.8\}$ and the crossover rate $cr \in \{0.2, 0.4\}$ were tested.

These settings were varied in the experiments and each one of the 192 possible configurations was tested ten times. All runs utilized a tournament selection scheme with five parents and were granted 10 000 generations. The following measurements were collected:

- ar.** The number of runs which found plans that completely covered all orders.
- at.** The *median* number of generations needed by the runs succeeding in this to find such plans.
- gr.** The number of runs which managed to find such plans which additionally were at least as good as the original freight plans.
- gt.** The *median* number of generations needed by the runs succeeding in this to find such plans.
- et.** The *median* number of generations after which f_2 did not improve by more than 1%, i. e., the point where the experiments could have been stopped without significant loss in the quality of the results.
- $e\tau$.** The *median* number of individual evaluations until this point.
- d.** The *median* value of f_2 , i. e., the median distance covered.

Table 1 contains the best ten configurations, sorted according to **gr**, **d**, and **$e\tau$** . The best configuration managed to reduce the distance covered by over

3000 km (17%) on a constant basis. Even the configuration at rank 170 (not in Table 1) saved almost 1100 km in median. 172 out of the 192 test series managed to surpass the original plans for the orders in the data set in ten out of ten runs and only eight configurations were unable to achieve this goal at all.

We furthermore applied significance tests (sign and Wilcoxon's signed rank test [26, 20]) in order to test which parameter settings have significant positive influence. On a significance level of $\alpha = 0.02$, we considered a tendency only if both tests agreed. Applying the convergence prevention mechanism (SCP) [20], larger population sizes, variety preserving fitness assignment [20], elitism, and higher mutation and lower crossover rates have all significantly positive influence in general.

One run of the algorithm (prototypically implemented in Java) for this data set takes around three hours, for 1000 orders it still fulfills the requirement of delivering result in 24 hours. Runs with 3000 orders, however, take much longer. These measurements were taken on a single dual-core 2.6 GHz machine, which is only a fraction of the capacity available in the dedicated data centers of the project partners. It is well known that EAs can be efficiently distributed and thus, the final implementation will be able to deliver results in time for all situations.

5.2 Tests with Multiple Datasets

We have run experiments with many other real-world order datasets for which the actual freight plans used by the project partners were available. In all scenarios, our approach yielded an improvement which was never below 2.3%, usually above 7%, and for some days even reaching areas above 15%.

Figure 3 illustrates the best f_2 -values (the total kilometers) of the individuals with the most orders satisfied in the population for two typical example evolutions. In the two diagrams, the total distance first increases as the number of orders satisfied by the solution candidates increases. At some point, all orders are satisfied and now, the optimization of f_2 begins to kick in fully. Soon afterwards, the efficiency of the original plans is surpassed.

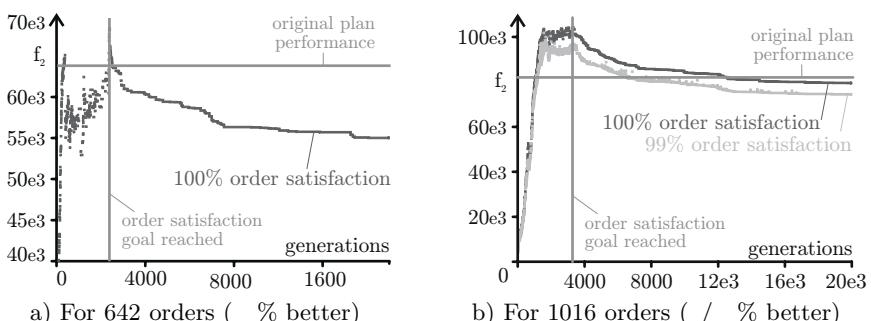


Fig. 3. Two examples for the freight plan evolution

6 Conclusions

In this paper we presented the *INWEST* freight planning component which utilized an evolutionary algorithm with intelligent reproduction operations. Although this issue has not been elaborated on, the system returns a full Pareto frontier of the planning problem to the human operator, who can then make her choice according to the given circumstances. Our approach was tested rigorously on real-world data from the *INWEST* partners and achieved excellent results.

In the current phase, the component was implemented rather prototypically. A re-implementation in a more efficient manner will most likely lead to speed-up of a few percent. Additionally, features like online updates of the distance matrix which is used to compute f_2 and by the genetic operators for determining the time a truck needs to travel from one location to another, are planned. The system will then be capable to *a)* perform planning for the whole orders of one day in advance and *b)* update smaller portions of the plans online if traffic jams occur. Then, the system will be deployed in the computer centers of the project partners.

References

1. Bundesministerium für Verkehr, Bau- und Stadtentwicklung: Verkehr in Zahlen 2006/2007. Deutscher Verkehrs-Verlag GmbH, Hamburg, Germany (2006)
2. Ceollo Coello, C.A., Lamont, G.B., van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, Heidelberg (2007)
3. CEN/TC 119: Swap bodies – non-stackable swap bodies of class C – dimensions and general requirements. EN 284, CEN-CEN ELEC, Brussels, Belgium (2006)
4. Glover, F.: Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13(5), 533–549 (1986)
5. Amberg, A., Domschke, W., Voß, S.: Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research (EJOR)* 124(2), 360–376 (2000)
6. Badeau, P., Gendreau, M., Guertin, F., Potvin, J.Y., Taillard, É.D.: A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies* 5(2), 109–122 (1997)
7. Bräysy, O., Gendreau, M.: Tabu search heuristics for the vehicle routing problem with time windows. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research* 10(2), 211–237 (2002)
8. Breedam, A.V.: An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints. PhD thesis, University of Antwerp, Belgium (1994)
9. Czech, Z.J., Czarnas, P.: Parallel simulated annealing for the vehicle routing problem with time windows. In: 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (PDP 2002), pp. 376–383. IEEE Computer Society, Los Alamitos (2002)
10. Bullnheimer, B., Hartl, R.F., Strauss, C.: An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research* 89, 319–328 (1999)

11. Doerner, K., Gronalt, M., Hartl, R.F., Reimann, M., Strauss, C., Stummer, M.: Savings Ants for the vehicle routing problem. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) *EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002*. LNCS, vol. 2279, pp. 11–20. Springer, Heidelberg (2002)
12. Jih, W., Hsu, J.: Dynamic vehicle routing using hybrid genetic algorithms. In: IEEE International Conference on Robotics and Automation, pp. 453–458 (1999)
13. Thangiah, S.R.: Vehicle routing with time windows using genetic algorithms. In: Practical Handbook of Genetic Algorithms: New Frontiers, pp. 253–277. CRC, Boca Raton (1995)
14. Zhu, K.Q.: A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. In: 15th IEEE International Conference on Tools with Artificial Intelligence, pp. 176–183. IEEE Computer Society, Los Alamitos (2003)
15. Alba, E., Dorronsoro, B.: Solving the vehicle routing problem by using cellular genetic algorithms. In: Gottlieb, J., Raidl, G.R. (eds.) *EvoCOP 2004*. LNCS, vol. 3004, pp. 11–20. Springer, Heidelberg (2004)
16. Ralphs, T.: Vehicle routing data sets, Data sets (2003) (accessed 2008-10-27), <http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/>
17. Pankratz, G., Krypczyk, V.: Benchmark data sets for dynamic vehicle routing problems (2007) (2008-10-27), http://www.fernuni-hagen.de/WINF/inhfrm/benchmark_data.htm
18. Weise, T., Zapf, M., Chiong, R., Nebro, A.J.: Why is optimization difficult? In: Nature-Inspired Algorithms for Optimisation. Springer, Heidelberg (to appear, 2009)
19. Radcliffe, N.J.: The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence* 10(4), 339–384 (1994)
20. Weise, T.: Global Optimization Algorithms – Theory and Application, 2nd edn. (2009) (accessed 2009-02-10), <http://www.it-weise.de/>
21. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford (1996)
22. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
23. Podlich, A.: Intelligente Planung und Optimierung des Güterverkehrs auf Straße und Schiene mit evolutionären Algorithmen. Master's thesis, Univ. of Kassel (2009)
24. Box, G.E.P., Hunter, J.S., Hunter, W.G.: Statistics for Experimenters: Design, Innovation, and Discovery. John Wiley & Sons, Chichester (2005)
25. Yates, F.: The Design and Analysis of Factorial Experiments. Imperial Bureau of Soil Science, Commonwealth Agricultural Bureaux, Tech. Comm. No. 35 (1937)
26. Siegel, S., Castellan Jr., N.J.: Nonparametric Statistics for The Behavioral Sciences. Humanities/Social Sciences/Languages. McGraw-Hill, New York (1988)

An Effective Evolutionary Algorithm for the Cumulative Capacitated Vehicle Routing Problem

Sandra Ulrich Ngueveu¹, Christian Prins¹, and Roberto Wolfler-Calvo²

¹ Institut Charles Delaunay - LOSI, Universit de Technologie de Troyes (UTT)
12, rue Marie Curie, 10000 Troyes, France
ngueveus@utt.fr, christian.prins@utt.fr

² Laboratoire d'Informatique de l'Universit Paris-Nord (LIPN)
99, av. Jean-Baptiste Clment, 93430 Villetaneuse, France
roberto.wolfler@lipn.univ-paris13.fr

Abstract. The Cumulative Capacitated Vehicle Routing Problem (or CCVRP) models transportation problems where the objective is to minimize the sum of arrival times at customers, taking into account capacity limitations. It generalizes the traveling repairman problem (or TRP), by adding capacity constraints and an homogeneous vehicle fleet. This paper presents the first metaheuristic designed for the CCVRP, taking into account specific properties to improve its speed and efficiency. The algorithm obtained also becomes the best metaheuristic for the TRP.

Keywords: Metaheuristic, routing for relief, cumulative capacitated vehicle routing problem, traveling repairman problem, minimum latency.

1 Introduction

The Cumulative Capacitated Vehicle Routing Problem (CCVRP) models transportation problems where the objective is to minimize the sum of arrival times at customers, taking into account capacity limitations. It belongs to the class of *vehicle routing for relief* problems which primarily aims at satisfying the customer need, e.g. vital goods supply or rescue after a natural disaster, instead of the classical route cost or length. It generalizes the traveling repairman problem (TRP), by adding capacity constraints and an homogeneous vehicle fleet.

Disaster relief logistics using routing approach has only been recently studied in literature [6], and to the best of our knowledge, only [4] addresses a problem equivalent to the CCVRP. The min-avg VRP it describes minimizes the average arrival time at customers. Applying an insertion heuristic on simple instances (unit demand, vehicle capacity modified, etc), the paper shows that good VRP solutions and min-avg VRP solutions can be significantly different.

More literature exist on the TRP, special case of the CCVRP, also known as the minimum latency problem, the delivery man problem or the school-bus driver problem. It consists in finding a tour of the set of nodes which minimizes the sum of the latencies to each node, where the latency of a node is its distance

along the tour. This problem is NP-complete, as shown by [11], and it cannot be considered as a simple variant of the classical traveling salesman problem (TSP) because of its specific properties on a metric space which makes it more difficult to solve or approximate, as explained in [3]. Most work on the TRP was on exact methods such as [3] or approximations algorithms such as [1]. The first and only metaheuristic published is a Greedy Randomized Adaptive Search Procedure with Variable neighborhood search (GRASP+VND) proposed in [12].

In the vehicle routing literature, the CCVRP differs from the VRP with min-max objective function [2], because the min-max VRP only minimizes the maximum arrival time at a customer, discarding the arrival times at the intermediate customers of each route, which can vary considerably if many customers are served early or late on their route. The CCVRP also differs from the Cumulative Vehicle Routing Problem (or CumVRP [8]) where the energy-minimizing objective function sums up the vehicle load times the length of traversed edges.

This paper reports the first metaheuristic designed for the CCVRP, taking into account specific properties to improve its speed and efficiency. It is organized as follows. Section 2 presents the mathematical formulation, some useful properties and the lower bounds derived. Section 3 focusses on the memetic algorithm designed. Section 4 shows the computational results, before the conclusion.

2 Mathematical Formulation and Useful Properties

2.1 CCVRP Formulation

The CCVRP is defined on a complete directed, symmetric and weighted network $G = (V, A, W)$ where $V = \{0, \dots, n, n + 1\}$ is the node set (nodes 0 and $n + 1$ correspond to the depot and $V' = V \setminus \{0, n + 1\}$ to the set of customers), A is the arc set and a weight w_{ij} is associated to each arc $(i, j) \in A$ to represent the distance or traveling time between nodes i and j . Q is the vehicle capacity, R is the vehicle fleet size and each node $i \in V'$ has a demand q_i . The objective of the CCVRP is to identify a set of routes so that every customer is visited exactly once and the sum of the arrival times at customers is minimized. A route is defined as a circuit, starting and ending at the depot, in which the total demand serviced does not exceed vehicle capacity Q . Let x_{ij}^k be a binary variable equal to 1 if arc (i, j) is used in route k and t_i^k the arrival time at node i if it is visited by route k . Inspired from the work from [7] on the VRP with time windows, the CCVRP can be formulated with equations (1) to (9).

$$\min F = \sum_{k=1}^R \sum_{i \in V'} t_i^k \quad (1)$$

subject to

$$\sum_{i \in V} x_{ij}^k = \sum_{i \in V} x_{ji}^k, \forall j \in V', \forall k \in [1..R] \quad (2)$$

$$\sum_{k=1}^R \sum_{j \in V} x_{ij}^k = 1, \forall i \in V' \quad (3)$$

$$\sum_{i \in V'} \sum_{j \in V} x_{ij}^k q_i \leq Q, \forall k \in [1..R] \quad (4)$$

$$\sum_{j \in V} x_{0j}^k = 1, \forall k \in [1..R] \quad (5)$$

$$\sum_{i \in V} x_{i,n+1}^k = 1, \forall k \in [1..R] \quad (6)$$

$$t_i^k + w_{ij} - (1 - x_{ij}^k)T \leq t_j^k, \forall i \in V, \forall j \in V, k \in [1..R] \quad (7)$$

$$t_i^k \geq 0, \forall i \in V, \forall k \in [1..R] \quad (8)$$

$$x_{ij}^k \in \{0, 1\}, \forall i \in V, \forall j \in V, i \neq j, \forall k \in [1..R] \quad (9)$$

The distinctive feature of this model is the objective function to be minimized: the sum of the arrival times at the different customers. Constraints (2) ensure the flow conservation: flow entering and exiting each node must be equivalent. Constraints (3) specify that each node is served by exactly one route. Constraints (4) are capacity constraints: the total load per route must not exceed vehicle capacity. Constraints (5) and (6) ensure that the depot is at the beginning and at the end of each route. Finally, constraints (7) compute the arrival times at each node and prevent subtours since T is defined as a large positive constant.

2.2 Useful Properties

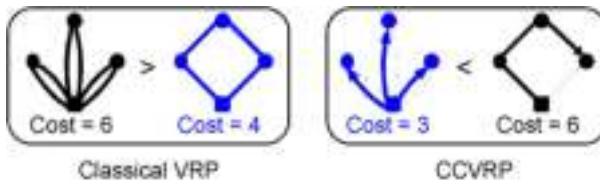
Because of their specific objective-function, customer-centric problems constitute an unusual class of routing problems. We present here some of the specific properties of the CCVRP, which show how different it is from classical routing problems. We managed to extract four very useful properties to improve the efficiency of our metaheuristic.

P1 : Classical Inequalities Are Not Valid for the CCVRP

Figure 1 illustrates one important difference between the CCVRP and the classical VRP. For a classical VRP, given a predefined number of vehicles, it is always better to send from the depot a single vehicle which visits all nodes of a given cluster before returning back to the depot. For the CCVRP, the opposite holds: if enough vehicles are available, it is always better to send directly one vehicle per node, so that each customer is served as soon as possible. This simple remark has a major impact on the problem: for example, the TRP does not produce a lower bound for the CCVRP.

P2 : TRP Does Not Produce a Lower Bound for the CCVRP

When the number of vehicles available $R \geq n$, then the optimal CCVRP solution is obviously obtained by connecting each node directly to the depot: each client is

**Fig. 1.** CCVRP vs VRP

served as soon as possible, with no “waiting time”. On the contrary, when $R < n$, then some clients must be regrouped in the same route to be served by the same vehicle, which delays some arrival times and therefore increases the objective function value. Consequently, solution values increase when R decreases and vice-versa. Therefore, although the TSP produces a lower bound for the VRP, the TRP cannot provide a lower bound for the CCVRP.

P3 : Optimal CCVRP Solutions Use Exactly $\min(R, n)$ Vehicles

Another consequence of the fact that CCVRP solution costs increase when R decreases and vice-versa, is that optimal CCVRP solutions will use all vehicles available. Indeed, any solution that uses less than R vehicles can be improved simply by removing the last node served by the fullest vehicle and assigning it to one of the empty vehicles. This property is the reason why our adapted splitting procedure described in section 3.1 extracts from each chromosome the best CCVRP solution of exactly R vehicles (assuming $R < n$).

P4 : Cost of a Reversed-Route: Data Pre-Processing

For given route k , let: n_k be the number of nodes of the route, c_i be the i^{th} node of the route and a_i be the i^{th} arc of the route: $a_i = (c_{i-1}, c_i)$. Therefore the arrival time t_{c_i} at the i^{th} node of the route is: $t_{c_i} = t_{c_{i-1}} + w_{a_i}$ and $t_{c_0} = t_{c_{n_k+1}} = 0$. Consequently, the cost F_k of this route can be expressed as follows:

$$\begin{aligned} F_k &= \sum_{i=0}^{n_k+1} t_{c_i} = t_{c_1} + t_{c_2} + \dots + t_{c_{n_k}} + t_{c_{n_k+1}} = w_{a_1} + (t_{c_1} + w_{a_2}) + \dots + \\ &(t_{c_{n_k-1}} + w_{a_{n_k}}) + 0 \\ &= w_{a_1} n_k + w_{a_2} (n_k - 1) + \dots + w_{a_{n_k}} \times 1 + w_{a_{n_k+1}} \times 0 \end{aligned}$$

$$F_k = \sum_{i=1}^{n_k+1} (n_k - i + 1) w_{a_i} \quad (10)$$

Let D_k be the total length of the route k (as computed for the classical VRP): $D_k = \sum_{i=1}^{n_k+1} w_{a_i}$. It can be easily demonstrated that if route k is reversed to obtain route rk , the cost F_{rk} of the new route is:

$$F_{rk} = n_k D_k - F_k \quad (11)$$

Equation (11) is key to design an efficient metaheuristic for the CCVRP. Since a route cost depends not only on the arcs used but also on the order of visit of

the nodes and arcs, the MA has to assess very quickly if a chosen route should be reversed or not. Equation (11) allows us to perform this test in $O(1)$ provided that the route cost and length have been pre-computed. We extend this idea in 3.3 to identify which data should be continuously stored and updated to be able to evaluate the impact of local search moves in $O(1)$.

2.3 Lower Bounds

LB1 : Unrestricted Vehicle Fleet Size

As explained in 2.2, solution values increase when R decreases and vice-versa. We also know that the optimal solution when $R \geq n$ consists in connecting each node directly to the depot. Therefore a simple lower bound *LB1* is:

$$LB1 = \sum_{j \in V'} w_{0j}.$$

LB2 : Arcs Cost Approximation

Let t_i be the i^{th} shortest arc of the graph. Considering equation (10), a lower bound $LB2_k$ of F_k can be computed as follows: $F_k = \sum_{i=1}^{n_k+1} (n_k - i + 1)w_{t_i}$. Therefore to compute $LB2 = \sum_{k=1}^R LB2_k$, all arcs are sorted in order of increasing weight then are assigned progressively to the lower position available in each route until all positions are filled: edge 1 goes into route 1 at position 1, edge 2 goes into route 2 at position 1, ..., edge R goes into route R at position 1, edge $R + 1$ goes into route 1 at position 2, ...

- When $R = 1$, $LB2$ becomes similar to the lower bound proposed in [12].
- $LB2$ can be improved by splitting the arc set A into A_0 and \bar{A}_0 as follows: $A_0 = \{(0, i) : i \in V'\}$, $\bar{A}_0 = \{(i, j) : i \in V', j \in V', i < j\}$. When computing $LB2$, the 1^{st} edge of each route must chosen from A_0 , and the remaining ones from \bar{A}_0 .

3 Memetic Algorithm

Memetic algorithms (MAs) belong to the class of evolutionary algorithms that hybridize local search within a classical genetic algorithm framework. We implemented an incremental MA which can be summarized as follows. We start with a population of σ chromosomes, representing CCVRP solutions and structured as explained in 3.1. At each iteration, two parents are selected after a binary tournament and the crossover described in 3.2 is applied to identify 8 potential children that can be obtained from these parents. The best child replaces a chromosome randomly selected among the 50% worst individuals of the current population. Every λ iterations, local search is applied to the best child before its insertion within the population. The algorithm stops after a predefined number of iterations *MaxIt*.

Algorithm 1. Memetic Algorithm

```

1: Initialize the population  $Pop$  of  $\sigma$  chromosomes
2: for  $It = 1$  to  $MaxIt$  do
3:   Select two parents  $A, B$  from  $Pop$  using binary tournament
4:   Apply crossover to  $A, B$  to get a set  $C$  of  $x$  children evaluated with split
5:   Select a random number  $rand$  in  $[0, 1]$  and define  $u := 0$ 
6:   repeat
7:      $u := u + 1$ 
8:     if  $rand \leq p_{ls}$  then
9:        $S :=$  the CCVRP solution found by split on  $C_u$  (the  $u^{th}$  fittest child)
10:      Apply local search on  $S$  and then convert  $S$  into chromosome  $C_u$ 
11:    end if
12:    Select a chromosome  $E$  in the worst half of  $Pop$ 
13:    if  $(F(C_u) < F(Pop(1)))$  or  $(|F(y) - F(C_u)| \geq \Gamma, \forall y \in Pop \setminus \{E\})$  then
14:      Replace  $E$  by  $C_u$ 
15:    end if
16:    until  $(u = x)$  or (Replacement performed)
17: end for

```

3.1 Chromosomes, Initial Population and Adapted Split

We use a classical chromosome structure and an adapted splitting procedure. Chromosomes are simple permutations of the n clients, without trip delimiters. Each of them has a *fitness* value (F), which corresponds to the cost of the best CCVRP solution extractable from the chromosome. Our adapted extraction procedure derives from the optimal splitting procedure of [9]. An auxiliary graph is built, in which each arc models a feasible trip (subsequence of chromosomes) and the optimal VRP solution (subject to the sequence) corresponds to shortest path of the graph. Its computation is done with Bellman's algorithm for acyclic graphs. The adapted procedure for the CCVRP uses the general form of this algorithm with the following invariant: at each iteration k , shortest paths with at most k arcs are identified. It stops at iteration R to obtain the CCVRP solution with at most R vehicles. Thanks to property P3, this procedure finds the best CCVRP solution using exactly R vehicles.

MA uses a population of σ chromosomes which is initialized with randomly generated elements except for two. The first individual is the solution of the nearest neighbor heuristic while the second is obtained after applying local search to it. To avoid clones within the population, the costs of any two solutions must be spaced at least by $\Delta > 0$. This criterion provides a sufficient dispersal and there is no need for a mutation operator.

3.2 Adapted Crossover

A classical order crossover OX as explained in [9] produces $x = 2$ children from 2 parents. We also test whether better children could be obtained after either one or both parents have been reversed. In total $x = 8$ potential children are

identified at each iteration and only the fittest is chosen to replace one of the 50% less fit chromosomes of the population, randomly selected.

3.3 Focus on the Local Search

Local search is applied to the child selected, with a probability p_{ls} , before its insertion within the population. The neighborhood is explored with 3 types of moves: classical 2-opt moves, node swap and node relocation. 2-opt moves are applied between 2 routes or within a single route. A node swap exchanges 2 nodes from the same or different routes. A node relocation removes a node from its current position or route to insert it at another position or route. In addition to these 3 classical moves, we test at each iteration whether one or both routes involved should be reversed or not.

Thanks to the property $P4$, the impact of such operation can be evaluated in $O(1)$, provided the route cost and length have been pre-computed. Formulas similar to equation (11) were derived to compute the cost variation of each local search move in $O(1)$ too, despite the fact that some moves reverse a portion of the routes involved.

4 Computational Evaluation

The computational evaluation was performed on the vrpnc instances of 50-199 nodes with no route length restriction proposed in [5] and available on the website [10]. For the CCVRP, we considered that the fleet size R would be the minimum number of vehicles necessary to handle the capacity constraints (like in classical VRP). We also used the TRP instances from [12] to compare our MA with their GRAPS+VND, only metaheuristic available on the TRP. All algorithms were coded in C on a Pentium D computer at 3.40 GHz with 1 GB of RAM running on Windows XP. We used a population of $\sigma = 15$ chromosomes during at most $MaxIt = 20000$ iterations, with a local search probability $p_{ls} = 0.30$.

In all tables of results, n , R and T show the number of nodes, the number of vehicles and the running times in seconds. In the table 3, column UB^* represents the best known upper bound. $LB1\% = \frac{100 \times LB1}{UB^*}$ and $LB2\% = \frac{100 \times LB2}{UB^*}$ show efficiency of the lower bounds 1 and 2 whereas $LB\% = \max(LB1\%, LB2\%)$. $NN\% = \frac{100 \times (NN - UB^*)}{UB^*}$ displays the results of the nearest neighborhood heuristic. $MA2\% = \frac{100 \times (MA2 - UB^*)}{UB^*}$ and $MA1\% = \frac{100 \times (MA1 - UB^*)}{UB^*}$ report the results of the memetic algorithm with and without parents reversed during crossover, T_{MA2} and T_{MA1} show the respective running times in seconds. In tables and , NbI represents the number of instances whereas $UB^+\% = \frac{100 \times (NN - MA)}{NN}$ shows the improvement over the nearest neighborhood heuristic.

4.1 TRP

As illustrated in Tables 1 and 2, our algorithms perform very well on the TRP, although they were not specifically designed for it. In the paper [12], two versions

Table 1. Deep algorithms on TRP

			deep GRASP + VND (1)			MA1 (2)		
NbI	n	R	UB ⁺ %	LB%	T	UB ⁺ %	LB%	T
20	10	1	2.44	33.04	0.00	2.43	73.32	0.00
20	20	1	9.86	39.63	0.04	10.11	67.51	0.01
20	50	1	9.67	45.41	3.21	9.36	62.42	1.44
20	100	1	11.56	43.40	101.27	11.95	63.52	93.26
20	200	1	11.33	42.32	3741.05	12.81	65.43	938.16
20	500	1	8.11	50.73	91470.78	13.85	63.69	16208.7

(1) From Salehipour, A. et al. (2008)

(2) Our MA with crossover OX + Reverse of none, one or both parents

Table 2. Fast algorithms on TRP

			fast GRASP + VND (1)			MA2 (2)		
NbI	n	R	UB ⁺ %	LB%	T	UB ⁺ %	LB%	T
20	10	1	2.14	33.48	0.00	2.43	73.32	0.00
20	20	1	8.19	42.43	0.00	10.11	67.51	0.01
20	50	1	6.85	50.07	0.27	9.32	62.42	0.84
20	100	1	10.24	45.53	9.73	11.92	63.52	37.96
20	200	1	10.69	43.37	356.73	12.70	65.43	354.58
20	500	1	10.70	46.49	7192.94	13.44	63.69	5365.52

(1) From Salehipour, A. et al. (2008)

(2) Our MA with crossover OX

of their GRASP+VND metaheuristic were presented: a *fast* version and a *deep* version which produces better upper bounds but is more time consuming. We did something similar: our MA2 has a simple crossover OX while our MA1 uses a crossover OX + reverse of none, one or both parents. We converted our running times to allow a fair comparison of our MA1 against their deep GRASP+VND and our MA2 against their fast GRASP+VND.

Table 1 shows that our MA1 outperforms the *deep* GRASP+VND: it is faster and gives better results. Table 2 shows that our MA1 produces better upper bounds than the *fast* GRASP+VND. In terms of running times, both metaheuristics are very fast on small instances (10-20 nodes), our MA2 is slower than GRASP+VND on medium instances (50-100 nodes) but faster on the largest instances (200-500 nodes). Overall we can conclude that our MA performs well and is currently the best metaheuristic available on the TRP although it was not specifically designed for it.

4.2 CCVRP

Both versions of our MA can be considered efficient on the CCVRP because they improve the nearest neighborhood heuristic by more than 30% and they are both less than 1% far from UB^* . As illustrated in Table 3, MA1 is 2.60 times faster than MA2, but the results of MA2 are 3.76 times better. The simple lower

Table 3. Results on the CCVRP

Pb	<i>n</i>	<i>R</i>	<i>UB</i> *	<i>LB1%</i>	<i>LB2%</i>	<i>NN%</i>	<i>MA1%</i>	<i>T_{MA1}</i>	<i>MA2%</i>	<i>T_{MA2}</i>
1	50	5	2230.35	53.86	84.02	23.69	0.00	6.77	0.00	15.80
2	75	10	2426.12	74.83	76.73	39.79	3.61	4.25	0.18	59.77
3	100	8	4045.42	61.67	72.87	38.02	0.68	65.27	0.68	168.78
4	150	12	4987.52	73.79	71.41	32.64	0.21	109.30	0.18	1069.56
5	199	17	5817.75	82.58	66.78	39.39	0.70	971.36	0.69	1617.23
11	120	7	7317.98	83.62	36.43	22.07	1.36	97.31	0.00	270.72
12	100	10	3558.92	81.08	65.57	30.16	0.01	16.11	0.01	103.83
Average			73.06	67.69	32.25	0.94	181.48	0.25	472.24	

bounds are also satisfactory: in average $\max(LB1\%, LB2\%) = 79.24\%$ and more than half of the instances are provided with a lower bound which exceeds 80% of UB^* .

5 Conclusion

This paper presents the first metaheuristic for the Cumulative Capacitated Vehicle Routing Problem (CCVRP), some useful properties and two simple lower bounds. The CCVRP is a generalization of the traveling repairman problem (TRP) which models transportation problems where the objective is to minimize the sum of arrival times at customers, instead of the classical route length, during vital goods supply for example or rescue after a natural disaster. The metaheuristic proposed is an efficient memetic algorithm (MA) which combines adapted crossover and splitting procedures with the useful properties we identified and pre-computed data. The properties we present were key to derive formulas able to evaluate the cost variation of each local search move in $O(1)$, as well as the lower bounds.

Computational results on the CCVRP show that our algorithm perform well not only on the CCVRP, but is also the best metaheuristic on the TRP although it was not specifically designed for it. Meanwhile, our simple lower bounds reach 80% of the best upper bounds known. Future work includes the extension of our MA to other customer-centric problems such as the multiroute-CCVRP.

References

1. Archer, A., Levin, A., Williamson, D.P.: A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing* 5(37), 1473–1498 (2008)
2. Arkin, E.M., Hassin, R., Levin, A.: Approximations for minimum and min-max vehicle routing problems. *J. Algorithms* 59(1), 1–18 (2006)
3. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The minimum latency problem. In: *Proceedings of the twenty-sixth annual ACM Symposium on Theory of Computing (STOC 1994)*, pp. 163–171 (1994)
4. Campbell, A.M., Vandenbussche, D., Hermann, W.: Routing for relief efforts. *Transportation Science* 42(2), 127–145 (2008)

5. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, L. (eds.) Combinatorial Optimization, pp. 315–338. Wiley, Winchester (1979)
6. Hsueh, C.-H., Chen, H.-K., Chou, H.-W.: Dynamic vehicle routing for relief logistics in natural disasters. In: Caric, T., Gold, H. (eds.) Vehicle Routing Problem, vol. 1, pp. 71–84. IN-TECH, Croatia (2008)
7. Kallehauge, B., Larsen, J., Madsen, O.B.G.: Lagrangean duality applied to the vehicle routing problem with time windows. Computers and Operations Research 33, 1464–1487 (2006)
8. Kara, I., Yetis, B.K., Yetis, K.: Energy minimizing vehicle routing problem. In: Dress, A.W.M., Xu, Y., Zhu, B. (eds.) COCOA 2007. LNCS, vol. 4616, pp. 62–71. Springer, Heidelberg (2007)
9. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 31, 1985–2002 (2004)
10. Reinelt, G.: (1995),
<http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>
11. Sahni, S., Gonzalez, T.: P-complete problems and approximate solutions. In: 15th Annual Symposium on Switching and Automata Theory, pp. 28–32. IEEE, Los Alamitos (1974)
12. Salehipour, A., Sörensen, K., Goos, P., Bräysy, O.: An efficient GRASP+VND metaheuristic for the traveling repairman problem. Working paper, University of Antwerp, Faculty of Applied Economics (May 2008)

A Corridor Method-Based Algorithm for the Pre-marshalling Problem

Marco Caserta and Stefan Voß

University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
`{marco.caserta,stefan.voss}@uni-hamburg.de`

Abstract. To ease the situation and to ensure a high performance of ship, train and truck operations at container terminals, containers sometimes are pre-stowed near to the loading place and in such an order that it fits the loading sequence. This is done after the stowage plan is finished and before ship loading starts. Such a problem may be referred to as pre-marshalling. Motivated by most recent publications on this problem we describe a metaheuristic approach which is able to solve this type of problem. The approach utilizes the paradigm of the corridor method.

1 Introduction

The Corridor Method (CM) is a novel metaheuristic recently proposed by [1]. The CM can be seen as a hybrid metaheuristic that bridges mathematical programming techniques with heuristic schemes. The basic idea of the CM is the use of an exact method over restricted portions of the solution space of a given problem. Given an optimization problem P , the basic ingredients of the method are a very large feasible space \mathcal{X} , and an exact method M that could easily solve problem P if the feasible space were not too large. Since, in order to be of interest, problem P generally belongs to the class of \mathcal{NP} -hard problems, the direct application of M for solving P usually becomes unpractical when dealing with real-world as well as large-scale instances.

The CM defines method-based neighborhoods in which a neighborhood is built taking into account the method M used to explore it. Given a current feasible solution $\mathbf{x} \in \mathcal{X}$, the CM builds a neighborhood of \mathbf{x} , say $\mathcal{N}(\mathbf{x})$, which can effectively be explored by employing M . Ideally, $\mathcal{N}(\mathbf{x})$ should be exponentially large and built in such a way that it could be explored in (pseudo) polynomial time using M . In this sense the CM closely relates to very large scale neighborhood search as well as the so-called dynasearch; see, *e.g.*, [2,3]. As the method is heuristic in nature, one could also substitute the concept of using an exact method M without fully sacrificing the genuine idea of the CM. Here one replaces M by an efficient heuristic and chooses a corridor in an appropriate way so that the heuristic is expected to work well on the chosen corridor.

Typically, a corridor around an incumbent solution is defined by imposing exogenous constraints on the original problem. The effect of such constraints is to identify a limited portion of the search space. The selection of which portions

of the search space should be discarded can be driven by a number of factors, *in primis*, by the power of the method itself deployed to explore the resulting neighborhood. However, one could also envision the design of a stochastic mechanism that, after dividing the search space in portions or limited regions, selects which of these subspaces should be included in the current corridor. Factors such as, *e.g.*, a greedy score, the cardinality of each subregion, etc., could be used to bias the stochastic selection mechanism that drives the definition of the corridor around the incumbent solution. The stochastic mechanism could be designed in such a way that, on the one hand, the corridor selection is non-deterministic, so that at every step different corridors around the same incumbent solution could be created and, on the other hand, such selection is still influenced by a merit score, that accounts for the attractiveness of each portion of the search space. Following such cooperative greedy stochastic corridor construction process, a balance between diversification and intensification is achieved, since, even though more promising regions of the search space have higher probabilities of being selected, not necessarily the best subregions will always be chosen.

The aim of this paper is to apply a metaheuristic approach based on ideas of the CM to solve the pre-marshalling problem (see, *e.g.*, [4,5] for a survey on related literature), a problem taken from the container terminals realm. First, we introduce the problem and illustrate how the CM can be used to deal with it. Then, Section 3 provides details about the proposed algorithm for the pre-marshalling problem. Section 4 reports results of the algorithm on a benchmark instance from the literature as well as on 40 randomly generated instances. Finally, Section 5 offers some concluding remarks.

2 The Pre-marshalling Problem

As pointed out by [4,6], much of the traffic of goods today is carried out through the use of containers at container terminals/seaports. Therefore, the ability to efficiently handle containers at container yards is paramount in enhancing the level of service at ports. The berthing time of ships at ports is perhaps the most important measure of port container terminal performance [4,6].

In container terminals, it is common practice to store outbound containers in a yard before loading them into a vessel. To be able to face competition among terminals and to guarantee a high level of service, operators must reduce unproductive time at the port [4]. One way of reducing the time of loading operations is to pile up containers taking into account their priorities such that a container with higher priority is never placed below a container with low priority. While such operation is carefully planned with this goal in mind, the final layout of the bay is strongly influenced by the arrival order of, *e.g.*, unloading trucks to the yard. For this reason, in the final bay configuration, high priority containers can still be found below low priority ones. Similar issues arise in inbound operations.

In the literature, the problem of arranging containers to maximize efficiency is extensively discussed; see [4,5] for a comprehensive review as well as references. A possible strategy is to define a good initial configuration for the bay via

shuffling, whether it be at the port yard, in the ship stowage, etc. The idea here is that containers are reshuffled and prearranged with the objective of minimizing the number of relocation moves required later on to load/unload containers. One common problem in this area is known as *stowage planning*, where one is concerned with finding the final layout of a ship stowage with the objective of minimizing future relocation moves when unloading at all ports visiting in a given route [5]. A related problem, known as *pre-marshalling* problem, is focused on determining the final layout of a bay at a port yard such that no relocation moves are necessary to load the containers into a vessel. Most recently, this problem has been studied, *e.g.*, by [7] and [6].

The pre-marshalling problem can, therefore, be seen as the problem of reshuffling the initial bay configuration in order to obtain a bay in which the retrieval operation can be carried out without further relocations, while minimizing the total number of moves required to reshuffle. In this paper we make use of some basic assumptions presented in [7], namely:

1. Pre-marshalling is considered only within the same bay (consisting of a certain number of stacks and tiers). No intra-bay movement is considered.
2. Containers (also addressed as *blocks*) are assumed to be of the same size.
3. The loading preferences of containers are known, *i.e.*, precedences among containers are assumed to be given in advance.

3 The Corridor Method Based Algorithm

In this section, we provide a detailed description of the major ingredients of the algorithm. Let us first define the concept of *forced relocations*. Given a current bay configuration, as in Figure 1, the number of forced relocations is given by the number of blocks in each stack currently on top of a block with higher priority. Such blocks will necessarily be relocated, in order to retrieve the block with higher priority located below. For example, in Figure 1, the number of forced relocations is equal to 4, as indicated by the shaded blocks. Note that the number of forced relocations in a bay constitutes a valid lower bound of the minimum number of relocations required to complete the retrieval operation.

The goal of the pre-marshalling problem is to reshuffle containers with the minimum number of movements such that the final configuration of the bay has zero forced relocations. Therefore, the proposed algorithm terminates when the first of the following stopping criteria is reached: The current configuration has zero forced relocations or a maximum running time has been reached.

The proposed algorithm is made up of four iteratively repeated phases:

Corridor Definition/Selection. Given the incumbent configuration, we first need to identify a *target* block to be relocated. Therefore, we first compute the number of forced relocations for each stack i in the bay. We then use a *roulette-wheel* mechanism to randomly select a stack in the bay such that the probability of selecting a stack is proportional to the number of forced relocations within the stack itself. Let us indicate with i^* the target stack

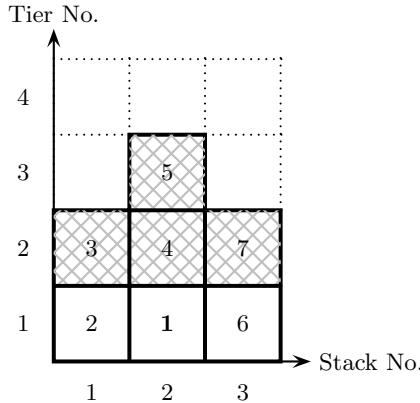


Fig. 1. An example of forced relocations within a bay. Blocks 3, 4, 5, and 7 will necessarily be relocated to retrieve blocks with higher priority.

and with t^* the uppermost block of the target stack. Consequently, let us describe the incumbent situation by (\mathcal{T}, i^*, t^*) , where \mathcal{T} represents the whole bay. The objective at each step of the algorithm is to relocate t^* from stack i^* to another stack, with the aim of reducing the number of forced relocations within the bay.

Once the target block t^* has been identified, a *corridor* around the incumbent solution is defined. The corridor width is influenced by the value of parameter δ , which indicates the number of stacks available for relocation. In order to reduce the size of the search space, we only allow relocations towards a subset of the available stacks, *i.e.*, only towards those stacks that are included in the current corridor. A stochastic mechanism is used in order to identify the subset of stacks to be included in the current corridor.

Given an incumbent configuration (\mathcal{T}, k, l) , we define a corridor around such configuration by selecting a subset of the stacks as admissible stacks. We identify three types of stacks with respect to the current configuration: (i) empty stacks, towards which it is always possible to relocate block l without generating any new forced relocation. Let us indicate with S_0 the set of such stacks in the current configuration; (ii) no-deadlock stacks, *i.e.*, stacks for which the block with highest priority has a lower priority than block l . In other words, if we indicate with $\min(\mathcal{T}_i)$ (in the following, $\min(i)$) the element with highest priority in stack i , then no-deadlock stacks are those stacks for which $\min(i) > l$. S_1 indicates the set of no-deadlock stacks; (iii) deadlock stacks, *i.e.*, stacks for which the block with highest priority has a priority higher than block l , $\min(i) < l$. In this case, relocating block l onto such a stack will generate a new relocation in the future, since we are going to retrieve one of the blocks in the stack prior to the retrieval of block l . S_2 indicates the set of deadlock stacks.

For each type of stack, a score is computed. The rationale behind such scores is that we try to capture the measure of attractiveness of each stack,

based upon whether a new deadlock is created after relocation of block l , and the impact of relocating block l onto a stack with respect to future relocations. The scores are as follows:

$$\sigma(i) = \begin{cases} \frac{1}{|S_0|}, & \text{if } i \in S_0 \\ \frac{\sum_{\substack{j \in S_1 \\ \min(j) \\ \min(i)}} \min(j)}{\sum_{j \in S_1} \min(j)}, & \text{if } i \in S_1 \\ \frac{\min(i)}{\sum_{j \in S_2} \min(j)}, & \text{if } i \in S_2 \end{cases} \quad (1)$$

Once the score of each stack is computed, we assign a weight to each category, say, w_0 , w_1 , and w_2 , such that $w_0 + w_1 + w_2 = 1$, and we normalize the scores such that $\sum_i \sigma(i) = 1$. Finally, a roulette-type mechanism is used to select which stacks belong to the corridor. The corridor selection scheme can easily be adapted to include, *e.g.*, only those stacks for which a maximum height, say λ , has not been reached.

Neighborhood Search. Given a set of admissible stacks Δ , we define a neighborhood of the current configuration (\mathcal{T}, i^*, t^*) by including in such a neighborhood only those configurations \mathcal{T}' that can be created relocating block t^* onto a stack $i \in \Delta$.

Move Evaluation and Selection. In order to evaluate each configuration, we define a greedy score. We employ a simple rule, given by the number of forced relocations (deadlocks) within the current configuration, where a forced relocation is imposed every time a block with higher priority is found below a block with lower priority. Let us associate with each configuration \mathcal{T} such greedy score $g : \mathcal{T} \rightarrow \mathbb{R}$, where $g(\mathcal{T})$ is equal to the number of forced relocations within configuration \mathcal{T} . We define the set of elite solutions in the current neighborhood Ω as the best 50% quantile of the overall set of solutions in $\mathcal{N}(\mathcal{T}, k, l)$. Finally, we employ a roulette-type stochastic mechanism to select one solution from Ω , in a fashion similar to what is presented in the corridor selection phase.

This step of the algorithm somehow resembles parts of what is known as semi-greedy heuristic [8] or GRASP [9,10]. At each iteration, the choice of the next element to be added is determined by ordering a set of candidate elements according to a greedy score. Such score measures the myopic benefit of selecting a specific move. The mechanism is adaptive in the sense that the score value associated to each candidate solution is changed to take into account the effect of the previous decisions. On the other hand, the roulette-type probabilistic mechanism allows to select different moves at each iteration of the algorithm, while still preserving a measure of attractiveness of each selection proportional to the value of the greedy score.

Note that we do not use the pure GRASP where the elements of the candidate set are randomly selected with equal probability, but with a score

dependent probability. While our idea is not necessarily a pure GRASP, it closely relates to, *e.g.*, the reactive GRASP mechanism of [11] or the classic fitness proportionate or roulette wheel selection mechanism known from genetic algorithms (see, *e.g.*, [12]).

Local Search Improvement. After reaching a new configuration, we begin the local search phase aimed at quickly improving the current configuration. The local search phase is made up by two different improvement schemes:

- the `heuristic_swap` scheme; and
- the `heuristic_subsequence_building` scheme.

The rationale behind the `heuristic_swap` scheme is to build sorted sequences in a stack, in such a way that stacks with zero forced relocations are defined. Figure 2 illustrates how the scheme works when an empty stack is found. The empty stack of Figure 2-(a) is iteratively filled up by identifying the uppermost block with highest value and relocating such block to the empty stack. The relocation sequence is, therefore, 11, 9, 8, 4 to stack 2, after which the swap operation is stopped, since the maximum height limit has been reached. Figure 2-(b) presents the final result of the swap operation.

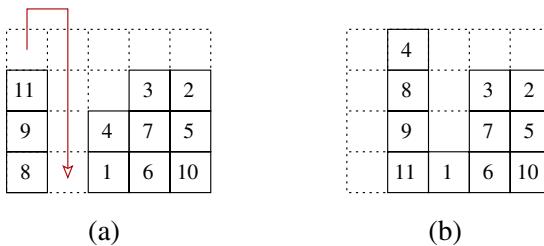


Fig. 2. Heuristic Swap Example: A sorted sequence is formed on an empty stack

A more complex swap operation is defined whenever there exists a stack with only one block on it. In Figure 3-(a), stack 2 is occupied by a single block, *i.e.*, block 2. We define a list \mathcal{L} of uppermost blocks with priority lower than 2, *i.e.*, $\mathcal{L} = \{11, 4, 3, 5\}$. If such list is not empty, we temporarily relocate block 2 on the stack that has the uppermost block with minimum value, among stacks with zero further relocations, if possible. In Figure 3-(a), we could place block 2 on stack 4, *i.e.*, on top of block 3 and, later on, place back block 2 to stack 2. However, since we could place block 2 on stack 5 without generating any new forced relocation, we move such block to stack 5. It is worthwhile observing that, since no new forced relocation has been created, we need not move back block 2 to stack 2 at the end of the swapping operation. Once stack 2 has been emptied, the same swapping operation described above in the case of an empty stack is applied. The final result of such swapping is presented in Figure 3-(b).

A third type of local search scheme is the `heuristic_subsequence_building` scheme. The idea is to build *tight* sequences, *i.e.*, sequences within the same stack

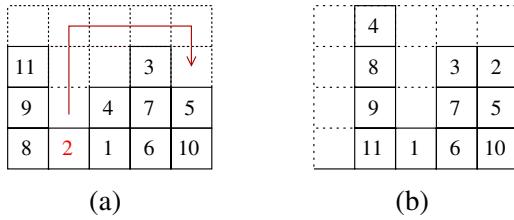


Fig. 3. Heuristic Swap Example: A sorted sequence is formed on a stack with only one block

made up by blocks (from top to bottom) whose priorities are $j, j + 1, j + 2$, etc. Therefore, we scan the whole bay and see whether an uppermost block t' could be moved from stack i' to stack i , where i has currently zero forced relocations. In order for the move to take place, we require that the uppermost block on stack i' has value $t = t' + 1$. For example, with respect to Figure 3-(b), a *tight* sequence could be built by relocating block 1 from stack 3 to stack 5, on top of block 2.

Finally, we exemplify an iteration of the proposed algorithm in Figure 4. In the figure, the four phases of the scheme, *i.e.*, (i) corridor definition, (ii) neighborhood design and exploration, (iii) move selection, and (iv) local search are identified. The four phases are iteratively repeated until one of the stopping criteria is reached. It is worth noting that the algorithm does not guarantee that a final solution is found. In fact, given a maximum height for the bay, the problem can become infeasible and, therefore, the algorithm terminates only when a maximum computational time is reached, without ever achieving a zero forced relocations configuration.

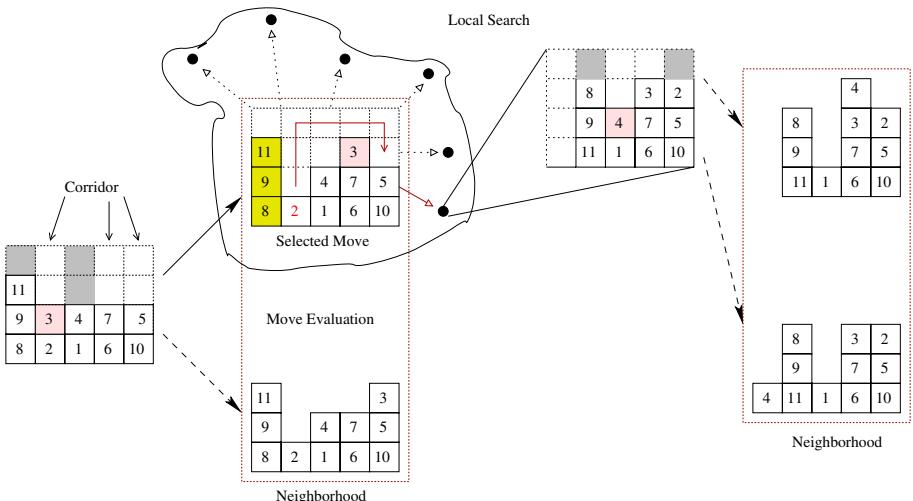


Fig. 4. The four-step algorithm: An iteration

4 Computational Experiments

In this section we present computational results on randomly generated instances as well as on a benchmark instance from the literature. All tests presented in this section have been carried out on a Pentium IV Linux Workstation with 512Mb of RAM. The algorithm has been coded in C++ and compiled with the GNU C++ compiler using the -O option.

The benchmark instance is taken from [6] and is a bay with 10 stacks and maximum height of 5 tiers. Other reasonably sized instances from [6] were not available. As in [6], stacks are numbered from 0 to 9, from left to right. There are 35 containers in the bay, with a utilization factor of 70%. Containers are divided in 10 classes, numbered from 0 to 9. Therefore, containers with the same priority belong to the same class. The initial layout is shown in Figure 5-(a).

(a)

2	5	0		2			5		
3	5	0	5	4			6		
3	5	0	6	5			6		
3	6	1	7	8		5	6		1
5	8	1	9	8	2	8	7	1	4

(b)

Fig. 5. Benchmark problem from [6]: (a) Initial layout, and (b) Final layout. The final layout is reached with 19 relocations.

Let us indicate with the triplet (k, a, b) a relocation of block k from stack a to stack b . The best movement sequence that we obtained with our algorithm is made up of 19 relocations (against the 31 relocations of [6],) i.e., $(4,8,4), (2,7,4), (7,2,7), (6,2,7), (6,6,7), (6,8,7), (5,2,7), (1,0,2), (0,1,2), (2,3,0), (5,3,1), (0,5,2), (5,9,1), (9,8,3), (5,5,1), (0,5,2), (7,6,3), (6,8,3)$, and $(5,9,3)$. The final layout is presented in Figure 5-(b). While our method is the one described above, the proposed heuristic of [6] consists of a neighborhood search process, an integer programming model, and three minor subroutines.

Table 1. Computational results on random instances. Each line is made up of 10 instances of the same size. Reported results are average values over 10 runs.

Bay Size		CM		Corridor	
h	m	No.	Time	δ^+	λ^*
3	3	21.30	20	2	$h + 2$
4	4	28.27	20	2	$h + 2$
5	5	46.91	20	3	$h + 2$
6	6	50.14	20	4	$h + 2$

$^+$: number of used stacks (not necessarily adjacent)

$*$: number of empty slots above the last filled block, *i.e.*, $\lambda = h + 2$.

The second part of the computational study is based upon an experiment that resembles that of [13,14] for a different problem. We used the same instances based on the data generation procedure of [13] also used in [14] for the blocks relocation problem and solved them in the spirit of the pre-marshalling problem. In Table 1, the first two columns define the instance size, in terms of tier and stack numbers. Columns three and four define the performance of the proposed algorithm, in terms of number of relocations and running time. Throughout the experiment, we fixed a maximum computational time of 20 seconds. This means that the algorithm was ran for 20 seconds over each instance, after which it was stopped, no matter whether a feasible solution was found. Finally, columns five and six provide information about the maximum corridor width, δ , and the maximum height for the bay, λ . We fixed $\lambda = h + 2$ for all instances. Values reported in the table are average values over 10 instances of the same size.

5 Conclusions

In this paper we have presented a novel algorithm based upon the Corridor Method paradigm. The algorithm is made up of four phases, in which ideas and features from the Corridor Method, GRASP or roulette wheel selection and basic local search techniques are effectively intertwined to foster intensification around an incumbent solution. From the first metaheuristic, the algorithm borrows the basic idea of building a corridor around the incumbent solution, by imposing exogenous constraints on the original problem. Conversely, the basic idea drawn from GRASP is the introduction of a stochastic construction process based upon a random selection of moves among a set of elite moves. Finally, basic local search operations are defined, *e.g.*, swap and move operations, to quickly improve the incumbent solution. The algorithm is stochastic in nature and based upon the use of a set of greedy rules that bias the behavior of the scheme toward the selection of the most appealing moves. The algorithm has been tested on a problem from the container terminal logistic domain, known as the pre-marshalling problem. In the pre-marshalling problem, container operators use “spare time” and “spare resources” to pre-arrange the bay in such a

way that no future relocations will be required during the subsequent retrieval phase. Besides other results, the best known solution of a benchmark instance from the literature has been considerably improved.

References

1. Sniedovich, M., Voß, S.: The Corridor Method: a Dynamic Programming Inspired Metaheuristic. *Control and Cybernetics* 35(3), 551–578 (2006)
2. Ergun, Ö., Orlin, J.B.: A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. *Discrete Optimization* 3, 78–85 (2006)
3. Potts, C., van de Velde, S.: Dynasearch - iterative local improvement by dynamic programming. Technical report, University of Twente (1995)
4. Steenken, D., Voß, S., Stahlbock, R.: Container terminal operations and operations research – a classification and literature review. *OR Spectrum* 26, 3–49 (2004)
5. Stahlbock, R., Voß, S.: Operations research at container terminals: a literature update. *OR Spectrum* 30, 1–52 (2008)
6. Lee, Y., Chao, S.-L.: A Neighborhood Search Heuristic for Pre-marshalling Export Containers. *European Journal of Operational Research* (2008), doi:10.1016/j.ejor.2008.03.011
7. Lee, Y., Hsu, N.Y.: An optimization model for the container pre-marshalling problem. *Computers & Operations Research* 34, 3295–3313 (2007)
8. Hart, J.P., Shogan, A.W.: Semi-greedy heuristics: An empirical study. *Operations Research Letters* 6, 107–114 (1987)
9. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP. Technical report, AT&T Labs Research (2004)
10. Binato, S., Hery, W.J., Loewenstern, D., Resende, M.G.C.: A greedy randomized adaptive search procedure for job shop scheduling. In: Ribeiro, C.C., Hansen, P. (eds.) *Essays and surveys in metaheuristics*, pp. 58–79. Kluwer Academic Publishers, Boston (2002)
11. Prais, M., Ribeiro, C.C.: Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing* 12, 164–176 (2000)
12. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading (1989)
13. Kim, K.H., Hong, G.P.: A heuristic rule for relocating blocks. *Computers & Operations Research* 33, 940–954 (2006)
14. Caserta, M., Voß, S., Sniedovich, M.: An algorithm for the blocks relocation problem. Working Paper, Institute of Information Systems, University of Hamburg (2008)

Comparison of Metaheuristic Approaches for Multi-objective Simulation-Based Optimization in Supply Chain Inventory Management

Lionel Amodeo^{1,*}, Christian Prins¹, and David Ricardo Sánchez²

¹ ICD, LOSI, University of Technology of Troyes,
12 rue Marie Curie, 10010 Troyes, France

lionel.amodeo@utt.fr, christian.prins@utt.fr

² Centro de Optimización y Probabilidad Aplicada Universidad de los Andes; A.A.
4976, Bogotá DC., Colombia
david-sa@uniandes.edu.co

Abstract. A Supply Chain (SC) is a complex network of facilities with dissimilar and conflicting objectives, immersed in an unpredictable environment. Discrete-event simulation is often used to model and capture the dynamic interactions occurring in the SC and provide SC performance indicators. However, a simulator by itself is not an optimizer. This paper therefore considers the hybridization of Evolutionary Algorithms (EAs), well known for their multi-objective capability, with an SC simulation module in order to determine the inventory policy (order-point or order-level) of a single product SC, taking into account two conflicting objectives: the maximization of customer service level and the total inventory cost. Different evolutionary approaches, such as SPEA-II, SPEA-IIb, NSGA-II and MO-PSO, are tested in order to decide which algorithm is the most suited for simulation-based optimization. The research concludes that SPEA-II favors a rapid convergence and that variation and crossover schemes play an important role in reaching the true Pareto front in a reasonable amount of time.

Keywords: Supply chain management, Simulation, Multi-objective Optimization, Metaheuristics, Evolutionary Algorithms.

1 Introduction

A Supply Chain (SC) can be defined as real world systems that transform raw materials and resources into end products which are consumed by customers. As each manufacturer is in charge of a subset of the SC that must be managed and run efficiently, the efforts to coordinate the members to act on behalf of the common good have led to the science known as Supply Chain Management (SCM). Our work deals with the SC operational issues of *inventory control* and management. Since the holding of inventories in an SC can cost between 20% and 40% of

* Corresponding author.

the value of the final product, deciding the *level of inventories* to be maintained as well as the *order-up-to level* in different stages of the SC is critical for the system total cost minimization and customer service level maximization. Because it is impossible to handle all the dynamically changing SC variables using analytical methods, *discrete-event simulation* is known as an effective analysis tool for dealing with stochastic variables as it can model multivariate non-linear relations. Nonetheless, in most cases, the study of the performance of the SC system goes in hand with the need to optimize the network performance. Alone, a simulation model is not an optimizer. But by introducing a *search procedure* that will communicate with the simulator which in turn evaluates the solutions given, both aims are achieved. This hybridization approach is known as *simulation-based optimization*, in which iteratively the output of the simulation is used by the optimization module to provide feedback on progress of the search for the optimal solution [6]. The early works on SC optimization focused on effective solutions for production planning [11] and mainly used heuristics and random generation of a large number of solutions, see for instance, Ettl et al. [5]. The use of EAs in the determination of an SC inventory policy was introduced by Daniel and Rajendran [2], who proposed a Genetic Algorithm (GA) to optimize the base-stock levels with the objective of minimizing the sum of holding and shortage costs in the entire supply chain. Köchel and Nieländer [8] used a simulation-based optimization method combined with a GA in order to find optimal order-points and order-quantities for all stages in the SC to minimize the total cost per year. Ding et al. [4] developed a simulation-based optimization module named ONE to support decision makers for the assessment, design and improvement of SC networks. Among strategic and tactical issues, inventory stock-levels were included in the search of the optimal Pareto front by the Non Dominated Sorting Genetic Algorithm (NSGAII). Finally, in Amodeo et al. [1], the performance of an SC is evaluated by simulating its Petri net model and, again, NSGA-II is used to guide the optimization search process toward high quality solutions of inventory policies. To the best of our knowledge, no study has been published to find the best metaheuristic technique for a simulation-based multi-objective optimization problem in SC inventory policy. Hence, the purpose of this paper is to test the coupling of different Evolutionary Algorithms (EAs) with a SC simulation module for the problem of finding the inventory policy in a SC with two conflicting objectives: total SC inventory cost and customer service level. Since each call to the simulator is time-consuming, the most adequate EA has to gain as much information as possible in every call to the simulation module.

2 Problem Formulation

In most cases, a SC is a serial linkage of interrelated echelons which perform a specific task leading to the final product. The interconnection can be complex and one echelon can take several roles. Generally, *suppliers* feed the *manufacturers* with raw materials for production to be carried out. Production is transferred by *distribution centers* to a final *retailer* who is responsible for selling the product

to a customer. However, in our case, the suggestion of a specific inventory policy is an operational level decision. Thus, given a specific SC design, sequential links between different sourcing, production and distribution activities or processes, we focus our attention on how the individual inventory policies can be defined. This way, the SC objectives can be achieved. As mentioned earlier, in view of one firm acting as an individual decision maker, the manager must determine what is the minimum inventory to be held to avoid stockouts (R : order-point) and which quantity to order at a time (Q : order-quantity). For the purpose of our study, the inventory revision is done periodically. This means that the stock level is checked in each period (day). If it is below the predetermined order-point (R), an order of size Q is placed in the preceding echelon. For a n facility SC, the *order point vector*: $R = (R_1, R_2, \dots, R_n)$ and the *order-quantity vector*: $Q = (Q_1, Q_2, \dots, Q_n)$ define the decision variables for the optimization procedure. R_i and Q_i refer to the i -th inventory reorder-point and reorder-level respectively. However, the echelons are interdependent. Each production facility replenishes the subsequent inventory and places orders on the preceding inventory in the SC. Nonetheless, if the order exceeds the stock of the preceding inventory, the order is partially filled and the unmet quantity is backordered until it is available in stock. This interconnection can be successfully modeled with discrete-event simulation. For the current investigation, the simulation module is based on the work [1]. This real SC is composed of three suppliers ($S1, S2, S3$); three upstream transporters (one for each supplier), a manufacturer ($S4$), its downstream transporters and a set of customers. All the echelons operate under a periodic-review base-stock policy, where the review time is one day. The manufacturer receives orders that randomly arrive with inter-arrival times subject to an exponential distribution with mean value 0.0355. If finished products are available, the mean customer transportation time is 2 days. The manufacturer needs three raw materials for the production of final products. They are purchased from suppliers 1, 2 and 3 respectively. The source of the supply of the raw materials to these three most upstream members is assumed to be infinite. However, there is a processing lead time (procurement/production/packing) and transportation lead time that are combined at each stage and considered as one component: average lead time. The mean replenishment lead time for the stocks $S1$ to $S4$ are 40, 20, 70 and 20 days respectively. For a given R and Q the two performance measures computed by simulation are: f_1 the *customer service level* to be maximized and f_2 the *total inventory cost* to be minimized. The service level is defined as the probability that customer orders can be satisfied on time. This is obtained by computing the fraction of time when the final retailer has orders but no ready inventory to process them. On the other hand, the inventory cost is calculated by taking into account the annual holding costs of raw materials and final products in stocks $S1, S2, S3, S4$ which are 0.3€, 0.6€, 0.16€ and 3€ respectively.

3 Multi-objective Optimization

The trade-off between customer satisfaction and inventory holding cost can be posed as a multi-stage stochastic optimization problem in which both order

quantity (Q) and the reorder point (R) are the key optimization variables. The goal is to compute a set of non-dominated or Pareto optimal solutions [13]. For our two objectives (1 - service level, 2 - inventory cost), a solution i dominates solution j ($i \succ j$) if $(f_1(i) \geq f_1(j))$ and $(f_2(i) < f_2(j))$ or $(f_1(i) > f_1(j))$ and $(f_2(i) \leq f_2(j))$.

3.1 SPEA-II

In this work, SPEA-II [12] is the first multiobjective GA which we adapted to our SC inventory policy problem.

Chromosome Representation: The representation used in this procedure is known in GAs as phenotype representation as it works directly with the actual values of the decision variables. One chromosome contains two substrings: $[R_1, R_2, \dots, R_n]$ for the order-point vector (R) and *Sub-string2*: $[Q_1, Q_2, \dots, Q_n]$ for the order-quantity vector (Q).

SPEA-II procedure:

1. Initialization: generate an initial population P_0 and create the initial external archive $\overline{P}_0 = \emptyset$; Set the generation counter, $t = 0$.
2. Fitness Assignment: calculate fitness values for individuals in P_t and \overline{P}_t .
3. Environmental Selection: copy all non-dominated individuals in P_t and \overline{P}_t to \overline{P}_{t+1} . If $|\overline{P}_{t+1}| > \overline{N}$ (archive size), then reduce \overline{P}_{t+1} by means of truncation operator, otherwise if $|\overline{P}_{t+1}| < \overline{N}$, fill \overline{P}_{t+1} with dominated individuals present in P_t and \overline{P}_t .
4. Termination: if $t \geq T$ (maximum number of generations), or stopping criteria are met, the set A (solution set) is represented by the non-dominated individuals in \overline{P}_{t+1} , stop. Otherwise go to next step.
5. Mating Selection: perform binary tournament selection with replacement on \overline{P}_t in order to fill the mating pool.
6. Variation: apply recombination and mutation operators to the mating pool and set P_{t+1} to the resulting population. Increment generation counter ($t = t + 1$) and go to 2.

Initialization: For each network site i , we are given lower and upper values $R_i^{\min}, R_i^{\max}, Q_i^{\min}, Q_i^{\max}$ for R_i and Q_i . Initial individuals are randomly generated with: $R_i = U[R_i^{\min}, R_i^{\max}]$; $Q_i = U[Q_i^{\min}, Q_i^{\max}]$; $\forall i = 1, 2, \dots, n$.

Fitness Assignment: Each individual in the archive \overline{P}_t and population P_t is assigned a strength value $S(i)$, representing the number of solutions it dominates: $S(i) = |\{j : j \in \overline{P}_t \cup P_t \wedge i \succ j\}|$. The raw fitness of the individual $Z(i)$ is calculated as: $Z(i) = \sum_{j \in \overline{P}_t \cup P_t \wedge j \succ i} S(j)$. And finally an element representing the closeness of solutions is included such that the fitness is given by $F(i) = Z(i) + D(i)$. $D(i)$ is calculated as follows where σ_i^k refers to the Euclidean distance between i and the k -th nearest neighbor: $D(i) = \frac{1}{\sigma_i^k + 2}$ where k is \sqrt{N} .

Chromosome Selection: Chromosomes are copied according to their fitness $F(i)$ into a mating pool. Binary tournament selection is used. Two individuals are randomly selected from population P_t . Individual with better fitness (lower $F(i)$) is chosen. This process is continued until the mating pool gets full.

Crossover and mutation: Single point crossover is utilized. A pair of children is created by exchanging parts of the two parents after the crossover point. The crossover point is a randomly selected integer. For mutation, *Gene Wise Mutation* [2] is used: each gene is modified as follows, where $x \in (0, 1)$ is a constant determined at the beginning of the run and $u \sim U[0, 1]$ is an uniform random number. $R'_i = (\min\{\max\{R_i(1 - x) + 2R_i \cdot x \cdot u, R_i^{\min}\}, R_i^{\max}\})$

Truncation: SPEA-II uses as density measure the σ_i^k . For archive truncation, the solution with the smallest σ_i^1 is eliminated. If two solutions share the same σ_i^1 value, the next nearest neighbor is compared until these values differ.

3.2 SPEA-IIb

We modified SPEA-II to obtain a new version, SPEA-IIb, aimed at taking advantage of the information gained by the simulator by introducing an aggressive reproduction scheme. It has also the purpose of improving the uniformity of the final Pareto-front by changing the original truncation operator. The general architecture of SPEA-II algorithm is maintained, and the only two changes are:

Crossover: the reproduction technique utilized is that from [2] referred to as *Variant II: Gene-wise*. Here, for one offspring, the information of as many parents as inventories in the SC is utilized. For our case, 4 parents are chosen at random and one offspring is created. The process is continued until the offspring pool is filled.

Archive Truncation: Crowding distance [3] is used as an archive truncation criterion. The crowding distance d_i of each solution in the external archive \overline{P}_{t+1} is calculated and the individual with smallest crowding distance is eliminated from the archive. To compute d_i , the archive is sorted in ascending order of f_1 . If i is the first or last solution in the sorted list, then $d_i = \infty$, otherwise $d_i = \frac{f_1(i+1) - f_1(i-1)}{f_1^{\max}(i+1) - f_1^{\min}(i-1)} + \frac{f_2(i+1) - f_2(i-1)}{f_2^{\max}(i+1) - f_2^{\min}(i-1)}$. Where $i-1$ and $i+1$ are the solutions before and after i and f_k^{\max} , f_k^{\min} , the *min* and *max* values of objective k .

3.3 Multi-objective Particle Swarm Optimization

The concept of Particle Swarm Optimization (PSO) is inspired by the flocking behavior of birds and was first proposed by Kennedy [7]. Several adaptations of the original version have been made in order to tackle problems with multiple objectives. We take the very recent implementation of Tripathi et al. [10] as it introduces a novel concept of time variant coefficients and can be easily adapt that work reasonably well. The general architecture of the algorithm is maintained, but flock representation is modified as follows:

Position Attribute: Stores the inventory policy $X = (R_1, Q_1, \dots, R_n, Q_n)$.

Velocity Attribute: Responsible for setting motion in each of the dimensions ruled by the Position Attribute. Such that V_{Ri} and V_{Qi} will update i -th inventory reorder-point and reorder-level respective positions $V = (V_{R1}, V_{Q1}, \dots, V_{Rn}, V_{Qn})$.

Experience Attribute: Stores the particle best individual policy $P = (P_{R1}, P_{Q1}, \dots, P_{Rn}, P_{Qn})$. The movement of the particle toward the optimum solution is governed by updating its position and velocity attributes [10]:

$$v'_{Ri} = w \cdot v_{Ri} + c_1 \cdot r_1(p_{Ri} - R_i) + c_2 \cdot r_2(p_{Rg} - R_i)$$

$$v'_{Qi} = w \cdot v_{Qi} + c_1 \cdot r_1(p_{Qi} - Q_i) + c_2 \cdot r_2(p_{Qg} - Q_i)$$

Where $w \geq 0$ is the inertia weight, $c_1, c_2 \geq 0$ the acceleration coefficients, and $r_1, r_2 \in [0, 1]$ are random numbers generated and responsible for providing randomness. The term $c_1 \cdot r_1(p_{Ri} - R_i)$ is called the cognition term and advocates just for the particle's individual experience. On the other hand, the term $c_2 \cdot r_2(p_{Rg} - R_i)$ accounts for a social interaction between particles.

MOPSO Procedure

1. Initialization: generate an initial flock F_0 and create the initial external archive $\overline{F}_0 = \emptyset$; Set generation counter, $t = 0$.
2. Evaluation: evaluate policy's performance proposed by flock F_t .
3. Update Personal Best: by comparing F_{t-1} best performance so far, calculate personal best for F_t .
4. Fitness Assignment: calculate fitness values of individuals in F_t .
5. Environmental Selection: copy all non-dominated individuals in F_t and \overline{F}_t to \overline{F}_{t+1} . If $|\overline{F}_{t+1}| > \overline{N}$ (archive size), then reduce \overline{F}_{t+1} by means of the truncation operator, otherwise if $|\overline{F}_{t+1}| < \overline{N}$, fill \overline{F}_{t+1} with dominated individuals present in F_t and \overline{F}_t .
6. Termination: if $t \geq T$ (maximum number of generations), stop, the set A (solution set) is represented by the non-dominated individuals in \overline{F}_{t+1} , stop. Otherwise go to next step.
7. Update: update acceleration coefficients and flock F_t velocity and position.
8. Mating Selection: perform binary tournament selection with replacement on \overline{F}_0 in order to fill the mating pool.
9. Variation: apply mutation operators to F_t and set F_{t+1} to the resulting population. Increment generation counter ($t \mapsto t + 1$) and go to 2.

Coefficient Update: Tripathi et al. [10] proposed time variant inertia and acceleration coefficients to be used with MOPSO. The idea of the time variant approach is utilized in order to have an adequate balance between local exploitation and global exploration. As a result, before the velocities and positions are updated, the acceleration coefficients are updated in the following way:

$$c_{1t} = \frac{(c_{1f} - c_{1i})}{g_{max}} + c_{1i} ; \quad c_{2t} = \frac{(c_{2f} - c_{2i})}{g_{max}} + c_{2i} ; \quad w_t = \frac{(w_f - w_i)}{g_{max}} + w_i$$

Where c_{1i}, c_{2i}, w_i refer to the social, local and inertia initial coefficient values respectively and c_{1f}, c_{2f}, w_f the final values which are specified at the beginning of the search and g_{max} represents the number of generations that will be used as stopping criteria.

Mutation: Tripathi et al. [10] suggest the inclusion of mutation. In our case, we include *Gene-Wise* mutation [2] which has already been explained in section 3.2.

4 Evaluation Criteria

When evaluating a multi-objective optimization technique, two complementary characteristics are often used: *Closeness*: solutions as close to the Pareto-Optimal solutions as possible and *Diversity*: solutions as diverse as possible in the obtained non-dominated front. Accordingly, two evaluation criteria are used:

1. *Coverage of two sets*: proposed by Zitzler [13]. Let $X', X'' \subseteq X$ be two sets of decision vectors. The function C maps the ordered pair (X', X'') to the interval $[0,1]$ as follows: $C(X', X'') := \frac{|\{a'' \in X'': \exists a' \in X': a' \succ a''\}|}{|X''|}$. The value $C(X', X'') = 1$ means that all points in X'' are dominated by or equal to points in X' . The opposite, $C(X', X'') = 0$, represents the situation when none of the points in X'' are covered by the set X' . Both $C(X', X'')$ and $C(X'', X')$ have to be considered as C is not symmetric.
2. *Spacing*: Schott [9] suggested a relative distance measure between consecutive solutions in the non dominated set Q , as follows: $S = \sqrt{\frac{1}{|Q|} \sum_{i=1}^{|Q|} (d_i - \bar{d})^2}$. Where $|Q|$ refers to the size of the non-dominated set and \bar{d} is the average of the d_i , which measures the distance between the solution i and the nearest solution in the objective space: $d_i = \min\{\sum_{m=1}^M (f_m^i - f_m^k)^2 | k \in Q \wedge k \neq i\}$.

5 Computational Experiments

SPEA-II, SPEA-IIb and MOPSO have been programmed on a Windows workstation using C++ under CodeBlocks environment. Results for NSGA-II come from the study [1] since they address the same particular SC setting. For the comparison, we use the same value for simulation parameters (horizon $T=2000$ days, $N = 25$ replications, warm-up period of 20 days). The computational time per simulation run is 0.78s per solution. The other parameters depend on the algorithm:

MOPSO specifications

- Social and local Coefficient: $c_{1i} = 2.5, c_{1f} = 0.5, c_{2i} = 0.5, c_{2f} = 2.5$.
- Inertia Coefficient: $w_i = 0.7; w_f = 0.4$; Mutation Probability: 0.1.

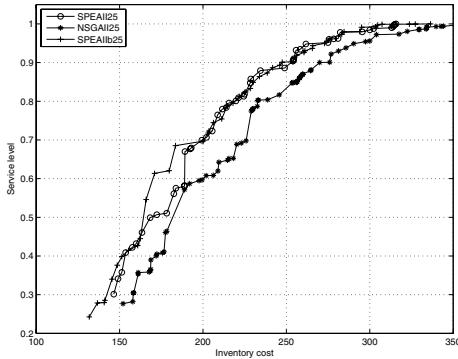
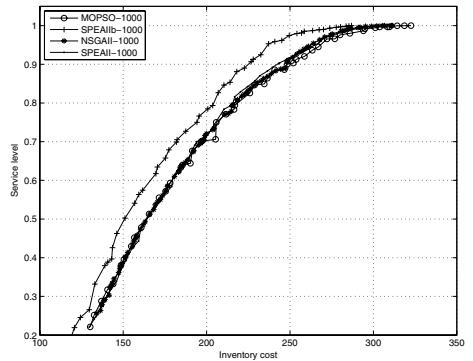
Table 2 shows Zitzler's measure $C(i, j)$ [12] where the comparison is made with NSGA-II to see if there is a significant improvement in comparison with the study in [1]. Figure 1 shows SPEA-II, SPEA-IIb and NSGA-II non-dominated

Table 1. Algorithm specifications

Mutation Probability	0.1	0.4	0.009091 ($\frac{1}{t}$)
Crossover	Single Point	Variant II: Gene-Wise	Variant II: Single Point
Crossover Probability	0.9	0.9	0.9
Size Archive	50	50	-
Size Population	100	100	100
Archive Truncation	σ_k [12]	Crowding Distance [3]	-

Table 2. Zitzler's measure with NSGA-II as reference

Generations	C(NSGA-II, i)					C(i,NSGA-II)				
	25	50	100	500	1000	25	50	100	500	1000
SPEA-II	0	0	2	36	32	100	100	64	24	26
SPEA-IIb	0	0	0	0	0	100	100	100	100	100
MOPSO	90	0	70	68	58	20.5	49	17	7	10

**Fig. 1.** SPEA-II, SPEA-IIb and NSGA-II output for 25 generations**Fig. 2.** Final Pareto-front of every algorithm tested

front for a 25 generation run. It is important to notice that results obtained with SPEA-II and SPEA-IIb completely dominate those obtained with NSGA-II. This is fundamental as we are looking for an EA that allows rapid convergence with little computational effort. Furthermore, if we compare the Pareto-front obtained with SPEA-IIb after 25 generations and NSGA-II after 1000, despite the fact that NSGA-II computed 975 more generations than SPEA-IIb, a first glance indicates that the non-dominated fronts are similar. It is even more astonishing to observe that some solutions in SPEA-IIb dominate those in NSGA-II. Although MOPSO shows good performance in early generation runs. Therefore, it must be said that MOPSO is not yet competitive with the established MOGAs. In any case, it remains it is a promising start for a first adaptation to this kind of problem and could be a basis for further work in hybrid simulation-optimization

Table 3. Schott's spacing measure and computational time

Generations	Spacing measure							
	25	50	75	100	250	500	750	1000
SPEA-II	3.137	1.332	1.521	1.400	1.233	1.293	1.500	1.550
SPEA-IIb	1.656	2.091	2.034	1.627	1.407	1.754	1.501	1.403
NSGA-II	1.993	1.385	1.157	1.380	0.802	0.814	0.874	0.790
MOPSO	3.423	3.304	2.447	2.053	1.372	1.334	1.456	1.411
Computational time (s)								
SPEA-II	396	828	1269	1706	4315	8520	12711	17080
SPEA-IIb	431	872	1306	1713	4066	7964	11907	15849
NSGA-II	1416	2758	4190	5543	13580	27132	41107	55762
MOPSO	440	859	1285	1701	4278	8683	13086	17485

metaheuristics in SC. Figure 2 shows the final Pareto-front approximation of each EA included in this study for a 1000 generation run. Clearly, and supported by the results in Table 2, SPEA-IIb's output approximates better to the real Pareto-front. Even though SPEA-II and SPEA-IIb early generations were very similar in terms of proximity, the modifications made in SPEA-IIb allowed for further exploration in the decision space, which resulted in outstanding results for the subsequent runs. Spacing measure for NSGA-II is clearly superior (Table 3) but for a decision maker it is much more useful to have the output generated by SPEA-IIb, as it improves the Pareto-front approximation.

6 Conclusion

In this study, we have investigated which EA can be best coupled with an SC discrete-event simulation module. It was done in order to benefit from EAs superiority as a multi-objective optimization search tool and to determine the SC inventory policy that will maximize customer service level while minimizing inventory cost. This is the first time in literature MOPSO has been used in an SC simulation based multi-objective optimization. Although MOPSO evidences a promising future, the verdict of the results forces one to temporarily discard MOPSO as the best alternative for the proposed problem. Revising our aim of finding the best EA in the hybrid approach, in order for the technique to be established as a reasonable decision decision tool, the time spent to reach the Pareto Front should be minimal. The results clearly show that the algorithm best suited to the simulator-optimization hybridization is SPEA-II with some modifications (SPEA-IIb). This can be concluded from two specific observations: SPEA-IIb or SPEA-II can reach the same Pareto-Front faster than NSGAI and for the final 1000 generation run, the difference between SPEA-II and SPEA-IIb, NSGA-II, MOPSO is extremely significant as SPEA-IIb output better approximates the actual Pareto-front. Thus, it can inferred that, because the general structure of the SPEA-II was not modified for SPEA-IIb evaluation, the internal functionality of Zitzler et al.'s [12] is the most appropriate as it allows a rapid convergence. However, the difference between the two approaches (SPEA-II and

SPEA-IIb) lies on an appropriate diversification technique. Due to the fact that the evaluation of a new solution consumes a considerable amount of time, the hybridized algorithm should be able to fully exploit all the information it has gained. As matters stand, aggressive reproduction schemes, such as the adapted *Gene Wise* reproduction scheme, with relatively high mutation probabilities, reveal significant results.

References

1. Amodeo, L., Chen, H., El Hadji, A.: Multi-objective supply chain optimization: An industrial case study. In: Giacobini, M. (ed.) *EvoWorkshops 2007*. LNCS, vol. 4448, pp. 732–741. Springer, Heidelberg (2007)
2. Daniel, J.S., Rajendran, C.: A simulation-based genetic algorithm for inventory optimization in a serial supply chain. *International Transactions in Operational Research* 12, 101–127 (2005)
3. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) *PPSN 2000*. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
4. Ding, H., Benyoucef, L., Xie, X.: A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization. *Artificial Intelligence* 19, 609–623 (2006)
5. Ettl, M., Feign, G.E., Lin, G.Y.: A supply network model with base-stock control and service requirements. *Operations Research* 48, 216–232 (2000)
6. Gosavi, A.: *Simulation-based optimization: parametric optimization techniques and reinforcement learning*. Kluwer Academic Publishers, Boston (2003)
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948. IEEE, Los Alamitos (1995)
8. Köchel, P., Nieländer, U.: Simulation-based optimization of multi-echelon inventory systems. *Int. Journal of Production Economics* 93–94, 505–513 (2005)
9. Schott, J.R.: Fault tolerant design using simple and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Boston, MA (1995)
10. Tripathi, P.K., Bandyopadhyay, S., Pal, S.K.: Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences* 177, 5033–5049 (2007)
11. Voss, S., Woodruff, D.L.: *Introduction to Computational Optimization Models for Production Planning in a Supply Chain*, 2nd edn. Springer, Berlin (2006)
12. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Zurich, Switzerland (2001)
13. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3, 257–271 (1999)

Heuristic Algorithm for Coordination in Public Transport under Disruptions*

Ricardo García, Máximo Almodóvar, and Francisco Parreño

Mathematics Department, University of Castilla-La Mancha, Spain

Abstract. This paper deals with on-line coordination of public transport systems under disruptions. An on-line optimization model is proposed in order to support decisions about how to balance all the fleet of transit lines in the public transport system and also to minimize waiting time caused by disruption. A fast heuristic algorithm is developed for the on-line problem and a numerical study of the regional train network of Madrid is carried out.

Keywords: Disruption, On-line Heuristic Optimization, Public Transport.

1 Introduction

The use of new technologies is having a great impact in real time management and control in public transport systems. It allows the evaluation of the system state by calculating transport demand dynamically, locating the vehicles at all times, or detecting incidents. This information helps transport managers, after detecting incidents, to modify the system schedule and manage services in real time.

This paper deals with the problem of deciding in real time which vehicles of a public transport system are reassigned to a line which unexpectedly has a demand that exceeds its capacity. The set of transit lines may be operated by several public transport lines. The main objective is to minimize the waiting time on the whole public transport network.

The topic of coordination between public transport lines in case of disruptions has not been treated in the literature as far as we know. Although some studies of it have been done, the problem of solving interruptions automatically is a relatively new topic so the literature about it is small. A seminal paper on the coordination of air routes under disruptions is Carlson [1]. For the studied problem, the proposed algorithms only solve instances with very few trips and planes. The algorithms used require a great computational effort and thousands of seconds to solve instances with hundreds of trips and a dozen planes.

Most optimization models found in the literature and closely related to our main topic are in one of these categories:

* The research described in this paper was partly supported by project PT-2007-003-08CCPP, CEDEX, Ministerio de Fomento, Spain.

- **Schedule reoptimization in railway systems.** In this problem the requirement is to adapt the service timetable after some disruptions appear. A feasible new timetable is wanted, as similar to the off-line timetable as possible, but where the new departures will be later than the established ones. A remarkable effort in this field is the ARRIVAL research project from European Commission Sixth Framework Programme <http://arrival.cti.gr/>. A similar problem to ours is studied by Adenso-Díaz et al.[2] where the impact of delays in vehicle timetables is analyzed. An integer programming is used where decision variables set what a vehicle must do (or if it must stay in its line). An on-line heuristic procedure based on “backtracking” is proposed in order to explore the solution space. Medanic and Dorfman[3] set out a discrete events model and a strategy to produce an efficient schedule and rescheduling in case of disruptions for the same problem. Recently, Törnquist [4] has presented a heuristic to reassign traffic in Swedish national railways.
- **Departure control in metro rail terminus.** Flamini and Pacciarelli [5] address a scheduling problem which arises in real time management of a metro rail terminus. Basically, it is about routing incoming trains through the station and scheduling their departures with the objective of optimizing punctuality and regularity of the train service.
- **Vehicle rescheduling problem (VRSP).** This problem appears when a vehicle on a scheduled trip breaks down, one or more vehicles need to be rescheduled to serve the passengers on that trip with minimum operating and delay costs. VRSP can be defined as the dynamic version of the classic vehicle scheduling problem (VSP) where assignments are generated dynamically. VRSP is about reassigning vehicles in real time to this disrupted trip as well as to other scheduled trips with given starting and ending times. Quan et al.[6][7] consider modelling, algorithmic, and computational aspects of the single-depot VRSP(SDVRSP). They also develop a parallel synchronous auction algorithm to solve it.

With real large scale problems including a lot of transit lines, the number of possible solutions can be so large that the best human expert cannot consider even a small number of options. Without a support tool, the expert will always send the nearest vehicle to the disrupted line. Although this idea is suggested by common sense, it is not always the best decision for the global service level. The on-line nature of the problem implies that reassessments must be made quickly because they have a direct bearing on service quality. This is the main reason why a constructive heuristic algorithm has been developed to solve the proposed model.

The paper is structured as follows: in Section 2 a description of the problem and the proposed model are given, in Section 3 we describe the constructive heuristic algorithm to solve the problem, in Section 4 a practical example with regional train services of Madrid (Spain) is reported. Finally, in the last section, we present some conclusions and future research.

2 The Optimization Approach

The model considers that the line $\ell = 0$ has been disturbed at instant $t = 0$. This random incident (including recurrent congestion and special events) implies that demand is greater than the capacity offered by the transit line $\ell = 0$. There is a set of transit lines \mathcal{L} whose vehicles can be reassigned to the line $\ell = 0$ in order to increase its capacity.

The problem consists of deciding which vehicles from the set of lines \mathcal{L} , and in which instants, must be reassigned to the disturbed line $\ell = 0$ in order to minimize waiting time of the system.

Figure 1 shows a schema of a proposed approach. On-line optimization decides if a vehicle which has to go out from line ℓ at instant s is reassigned (variable value x_s^ℓ). These decisions are taken into account by a simulation model which updates the departure schedule for system vehicles, $\mathbf{t}(\mathbf{x})$, and finally the objective function allows the calculation of system total waiting time in real time, $W(\mathbf{t}(\mathbf{x}))$, dependent on the new timetable.

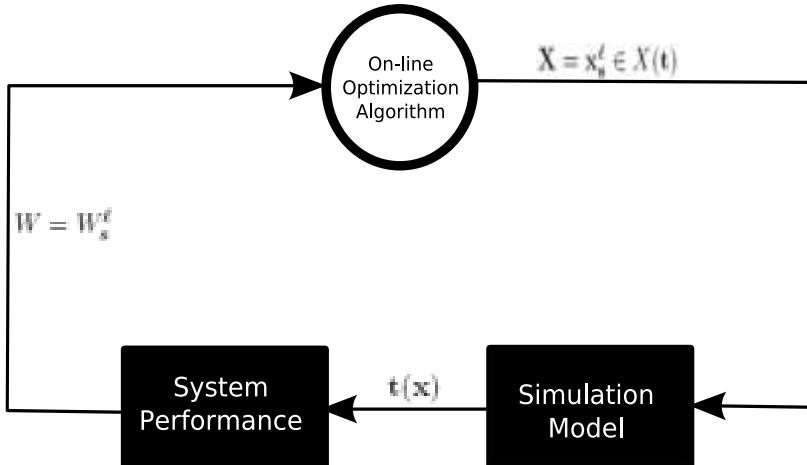


Fig. 1. The optimization approach

More formally, the problem tackled by this paper is formulated as the following optimization problem, where $X(\mathbf{t})$ is the set of instants in which a vehicle can be reassigned:

$$\begin{aligned} &\text{minimize} \quad F = W(\mathbf{t}(\mathbf{x})), \\ &\text{subject to:} \quad \mathbf{x} \in X(\mathbf{t}) \\ &\qquad\qquad\qquad x_s^\ell \in \{0, 1\} \end{aligned} \tag{1}$$

The essential difficulties of this problem are: i) the feasible set depends on decision variables, i.e. feasible modifications of timetables depend on previous modifications over them, so the restriction set cannot be described in an explicit way, ii) depending on the detail level adopted in the simulation model, computational cost can be high, and iii) It is a problem which must be solved in

real time. These considerations lead to the solving of the previous model with a heuristic approach.

2.1 Decision Variables

There are some assumptions for reassignment of vehicles:

Assumption 1. Transit vehicles can only be reassigned when they are located in one of the two headers, $p \in \{+, -\}$, of the line. This implies they go empty towards the disturbed line.

Assumption 2. No rescheduling is done when a vehicle is re-assigned.

Decision variables are x_s^ℓ which means what decision has been taken in line ℓ at instant s . Each value of this decision variable has a different effect:

Reassignment ($x_s^\ell=1$): It means that a vehicle which has to depart at instant t_s in line ℓ must go to the disturbed line $\ell = 0$, or if $\ell = 0$ it has to return to its original line before disruption appears in the system.

Non-reassignment ($x_s^\ell=0$): It means that the vehicle takes its scheduled departure.

2.2 Simulation Model

In the methodological schema we have considered, the simulation model estimates new timetables when variables x_s^ℓ modify programmed timetables. The proposed optimization algorithm takes the model as a “black-box”. This fact allows consideration of any generic model. In this article, the considered model operates with a periodic timetable under the following hypothesis:

Assumption 3. Vehicle speed is constant throughout the period of study.

Assumption 4. When a reassignment occurs, the time which a vehicle travelling from header to header takes does not depend on the instant t .

Figure 2 shows a pair of reassessments to the disturbed line. Initially, time between two consecutive departures, τ^ℓ , is constant. Reassessments make some irregularities in scheduled departures. Reassigned vehicles go to the disturbed header which minimizes time to enter service. For the disturbed line, time has been made discrete to have a minimum time, τ^0 , between two consecutive departures. Reassigned vehicles are incorporated into their incorporation header at the first possible instant. When it is decided that a vehicle has to return to its original line, the mechanism to decide into which header and in which instant it is incorporated is the same. But in this case, the instant matches with the original timetables of the original line.

In a simplified way, the simulation model considers a set of *potential instants* which is formed by all definitive departures in original timetables plus a discretization of the disturbed line $\ell = 0$. In this set of instants there is another subset of *decision instants* in which a vehicle departs in an effective way. Decision variables x_s^ℓ modify dynamically these decision instants within the set of

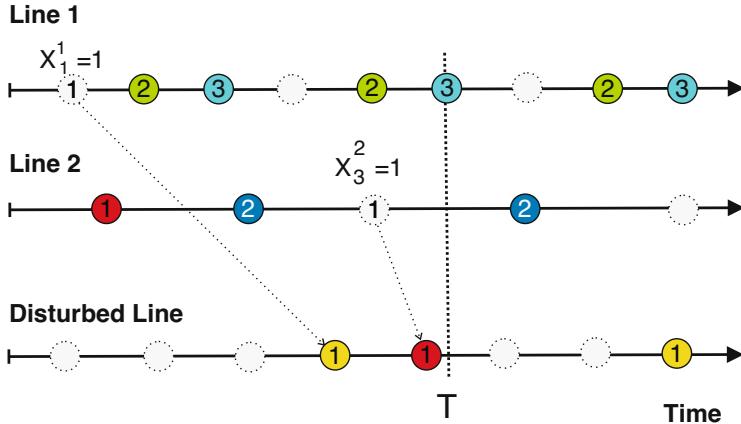


Fig. 2. System events $t(x)$

potential instants. In this paper, a subindex s is assigned to a generic element from the potential instant set. Due to space limitations, we omit the pseudo-code for this simulation.

2.3 Objective Function

The proposed model is intended to obtain a balance between service supply of the transport system and its demand when a disruption affects the system. Fundamental concepts for describing this balance are system supply capacity and service demand, both of which define a queue model. For each transport line ℓ there is a queue system formed by a set of individual queue systems, one per station. The set of vehicles and its characteristics define system capacity, and together with spatial distribution of demand (demand matrix of origin-destination trips between stations) and its temporal generation determine system queues. All these mechanisms can be considered by a microscopic simulation model. In this work, basically motivated by on-line characteristics of the problem, the following hypotheses have been assumed:

Assumption 5. The queue system for each line ℓ has been considered as a single element.

Assumption 6. Before disruption, demand has been satisfied in every line. So header queues are empty in both sides $p \in \{+, -\}$.

Dynamic modelling of demand requires the use of two concepts: i) flow intensity in each instant, for each line, and in each direction $v_p^\ell(t)$ in order to describe demand generation and ii) number of users who enter each line $N_p^\ell(t)$, shown in figure 3, intended to describe the size of the waiting queue. The relation existing between them is:

$$N_p^\ell(t) = \int_0^t v_p^\ell(t) dt \quad (2)$$

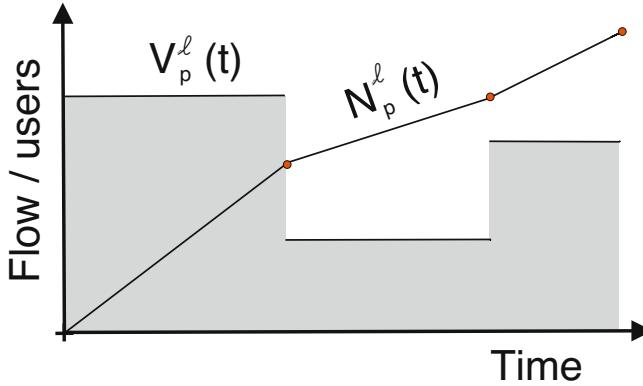


Fig. 3. Flow and number of users (demand) in a line ℓ

Equations for calculating queue size at decision instant s :

$$Q_s^\ell = \min \{Q_{s'}^\ell + N_s^\ell - N_{s'}^\ell - k_s^\ell, 0\} \quad (3)$$

$$N_s^\ell = N_p^\ell(t_s^\ell) \quad (4)$$

where s' is the last decision instant, p is the instant s header (+ or -), and $v_p^\ell(t)$ is the demand flow at time t for the line ℓ and direction p . k_s^ℓ represents number of users which can get on a vehicle which has its departure at instant s for line ℓ . This parameter depends on vehicle capacity, demand distribution (in which stations passengers get on and off) and finally on the critical line section/s. This parameter can be approximated using a microscopic simulation model executed off-line.

Using data obtained by the applications of these formulas, we can calculate the waiting time in each decision time. First, we should estimate the waiting time which a decision instant adds to total waiting time in line using

$$W_s^\ell = Q_{s'}^\ell (t_s^\ell - t_{s'}^\ell) + \int_{t_{s'}^\ell}^{t_s^\ell} v_p^\ell(t)(t_s^\ell - t) dt \quad (5)$$

The last expression can be calculated depending on whether demand changes during the period $[t_{s'}^\ell, t_s^\ell]$. When it does not change the formula can be simplified as in (6)

$$W_s^\ell = Q_{s'}^\ell (t_s^\ell - t_{s'}^\ell) + \left(\frac{N_s^\ell - N_{s'}^\ell}{2} \right) (t_s^\ell - t_{s'}^\ell) \quad (6)$$

Otherwise waiting time is calculated by using the compound trapezoidal integration rule to approximate the value of integral (5) as follows:

$$W_s^\ell = Q_{s'}^\ell (t_s^\ell - t_{s'}^\ell) + \frac{h}{2} \left[v_p^\ell(t_i)(t_s^\ell - t_{s'}^\ell) + 2 \sum_{i=1}^{m-1} v_p^\ell(t_i)(t_s^\ell - t_i) \right] \quad (7)$$

where $h = \frac{t_s^\ell - t_{s'}^\ell}{m}$, $t_i = t_{s'}^\ell + hi$ with $i = 1, \dots, m-1$ and m is a positive integer large enough to calculate (5) with enough precision.

To calculate the total waiting time in a line we have to add the waiting times for all decision instants, $W^\ell = \sum_s W_s^\ell$. Note that the final instant of the disruption period is considered as a dummy decision instant. So finally, the global waiting time is

$$W = \sum_{\ell \in \mathcal{L} \cup \{0\}} W^\ell \quad (8)$$

3 On-Line Optimization Algorithm

The on-line optimization algorithm simulates and optimizes the transport system simultaneously. It works as follows:

Optimization phase. A parameter Δt is chosen so that $\Delta t \leq \min_{\ell \in \{\mathcal{L} \cup 0\}} \{\tau^\ell\}$. Due to this choice, each line has (at most) only one instant to be considered inside the time period $[t, t + \Delta t]$. The algorithm consider vehicles which have their scheduled departure in period $[t, t + \Delta t]$. The benefit from reassigning a specific vehicle depends on the rest of the reassignment decisions taken in other lines. Due to this, the algorithm initially evaluates the individual benefit of each line to reorder decisions from best to worst. Later, with an established order, it recalculates reassignment decisions but assuming that previous decisions have been taken. The algorithm chooses from a reassignment list which reduce global time of the system.

Simulation phase. From decisions x_s^ℓ taken in optimizaton phase the algorithm simulates the transport system at period $[t, t + \Delta t]$. After this, the algorithm reiterates the process.

The ordering factor is based on a Winning - Losing formula using waiting times. The algorithm schema is shown on Table 1. The most important thing in the algorithm is estimation of the rightness of decision based on the formula

$$\bar{W}_t^\ell + \hat{W}_t^\ell - (W_t^\ell + W_t^0) > 0.$$

When we analyze a decision we must see the effects of it in the disrupted line and in the analyzed line. In addition we must consider total waiting times under both decisions: reassignment or maintenance. So we can define W_t^0 and \bar{W}_t^0 as total waiting time in the disrupted line under maintenance and reassignment respectively. In the same way, we can define $W_t^{\ell'}$ and $\hat{W}_t^{\ell'}$ as total waiting time in the analyzed line under maintenance and reassignment respectively. Our winning factor is $\bar{W}_t^0 + \hat{W}_t^{\ell'}$ which means the global waiting time under reassignment and our losing factor is $W_t^0 + W_t^{\ell'}$ that means the global waiting time under maintenance. Obviously we want the first global time to be less than the second one, so we need Winning - Losing > 0 . And from this we can extract the basic formula for rightness of decisions.

Table 1. On-line optimization constructive algorithm schema

-
0. **Initialization.** $t = t_0$ is the starting system instant where disruption happens. Let Δt obey $\Delta t \leq \min_{\ell} \{\tau^{\ell}\}$. Take $\mathcal{I}_t = \{\emptyset\}$
 1. **Optimization phase.** Let \mathcal{I}_t be a set of lines which have a decision instant in period $[t, t + \Delta t]$.
 2. For each $\ell \in \mathcal{I}_t - \{\emptyset\}$ calculate
 - W_t^0 : waiting time in line $\ell = 0$ since instant t (the same for everyone).
 - W_t^{ℓ} : waiting time in line ℓ since instant t .
 - \bar{W}_t^{ℓ} : waiting time in line ℓ since instant t but reassigning a vehicle from line ℓ whose departure falls in period $[t, t + \Delta t]$.
 - \hat{W}_t^{ℓ} : waiting time in disturbed line $\ell = 0$ since instant t under decision $x_{b\ell}^{\ell} = 1$ (reassigning vehicle from line ℓ whose departure falls in period $[t, t + \Delta t]$).
 3. Considering the next set

$$\mathcal{I}_t^+ := \left\{ \ell : \bar{W}_t^{\ell} + \hat{W}_t^{\ell} - (W_t^{\ell} + W_t^0) > 0 \right\}$$

Order the set from highest to lowest.

4. For all $\ell \in \mathcal{I}_t^+$, do:

Recalculate \hat{W}_t^{ℓ} .

If

$$\bar{W}_t^{\ell} + \hat{W}_t^{\ell} - (W_t^{\ell} + W_t^0) > 0$$

then reassign vehicle from line ℓ which has departure in period $[t, t + \Delta t]$, i.e. $x_{b_s}^{\ell} = 1$.

5. If $\ell = 0 \in \mathcal{I}_t$, being ℓ' original line attending disturbed line

Calculate

- W_t^0 : waiting time in line $\ell = 0$ since instant t .
- $W_t^{\ell'}$: waiting time in line ℓ' since instant t .
- \tilde{W}_t^0 : waiting time in line $\ell = 0$ since instant t with decision $x_{b0}^0 = 1$ (vehicle reassignment from line $\ell = 0$ which has departure in period $[t, t + \Delta t]$).
- $\tilde{W}_t^{\ell'}$: waiting time in line ℓ' since instant t with decision $x_{b0}^0 = 1$.

If

$$\tilde{W}_t^{\ell'} + \tilde{W}_t^0 - (W_t^0 + W_t^{\ell'}) > 0$$

take decision $x_{b0}^0 = 1$.

6. **Simulation phase.** With all decision taken in period $[t, t + \Delta t]$ simulate system until instant $t + \Delta t$. Take $t = t + \Delta t$ and return to Step 1.
-

4 Tests

We have done some numerical tests using the data from regional train services of Madrid. This problem has a small number of lines (10 lines) and the numerical tests use a simplified simulation model. Tests demonstrate the speed of our algorithm for this problem. In these numerical tests, the execution time was less than 5 minutes to simulate 50 minutes using an Intel Core2Duo E6850 at 3.00GHz. The execution depends on time and changes when it is done in a different instant. Our response also depends on the geographic situation of the disturbed line so, when this disturbed line is too far from the assisting lines, response time is a little longer.

Figure 4 shows the results of the average global waiting time in the set of lines \mathcal{L} and for the assisted line $\ell = 0$, as a function of the time t . This is $W^0(t)$ and $\frac{\sum_{\ell \in \mathcal{L}} W^\ell(t)}{|\mathcal{L}|}$

This graph shows how, with dynamical actions in the system, total expected time in line $\ell = 0$ can be reduced by incrementing waiting time in the rest of system \mathcal{L} . This problem has an interpretation as a balance game in which players (vehicles) change their strategy (line) and with it reduce total time in the system (benefit). In the solution to this game all costs tend to balance, otherwise players would change their strategy.

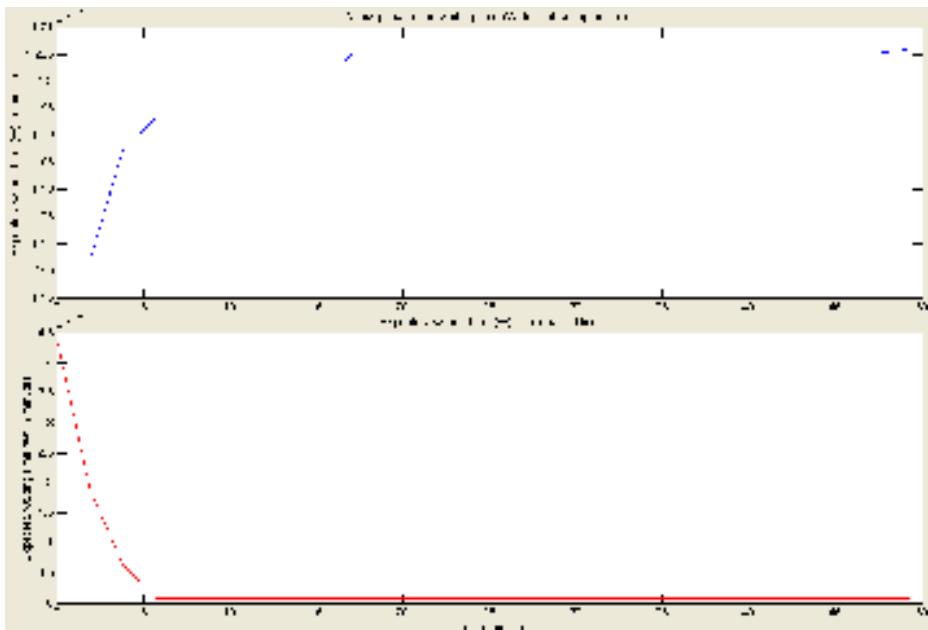


Fig. 4. Test on October 11th, 2008 at 07:30:45

5 Conclusions

This paper tackles the problem of deciding in real time how a set of public transport lines can help a line whose demand increases too much for its capacity. An optimization schema has been proposed based on: i) a predictive simulation model, ii) a queue system, and iii) a constructive heuristic algorithm.

Numerical tests carried out in the Madrid local railway network show the possibility of applying this on-line schema. However, in systems where the number of lines is high (like in buses), or where a microscopic simulation model is employed, computational cost can be high. The proposed heuristic algorithm has the possibility of parallelizing calculations of individual benefit in each line for hard cases.

At present, we are working on taking real time information by using a communication system between our approach and the real transport system. By taking real time data, the model can change, so simply by sending new information to the algorithm our system can be adapted to work online.

References

1. Carlson, P.M.: Exploiting the opportunities of collaborative decision making: A model and efficient solution algorithm for airline use. *Trans. Sci.* 34, 381–393 (2000)
2. Adenso-Díaz, B., González, O., González-Torre, P.: On-line timetable re-scheduling in regional train services. *Trans. Resrch. Part B* 33, 387–398 (1999)
3. Medanic, J., Dorfman, M.J.: Efficient scheduling of traffic on a railway line. *J. Optim. The. Appl.* 115, 587–602 (2002)
4. Törnquist, J.: Railway traffic disturbance management—An experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. *Trans. Resrch. Part A: Pol. and Prac.* 41(3), 249–266 (2007)
5. Flamini, M., Pacciarelli, D.: Real time management of a metro rail terminus. *Eur. Jour. Oper. Resrch.* 3, 746–761 (2008)
6. Li, J.-Q., Mirchandani, P.B., Borenstein, D.: The vehicle rescheduling problem: Model and algorithms. *Netw.* 50(3), 211–229 (2007)
7. Li, J.-Q., Mirchandani, P.B., Borenstein, D.: Parallel Auction Algorithm for Bus Rescheduling. *L. N. in Econ. and Mat. Sys.* 600, 281–300 (2008)

Optimal Co-evolutionary Strategies for the Competitive Maritime Network Design Problem

Loukas Dimitriou and Antony Stathopoulos

Department of Transportation Planning and Engineering, School of Civil Engineering,
National Technical University of Athens, 5 Iroon Polytechniou Str, 15773 Athens, Greece
lucdimit@central.ntua.gr, a.stath@transport.ntua.gr

Abstract. The current paper is focusing into the less well-defined transportation networks as those that are formed by the integration (combination) of alternative transportation means for servicing freight movements and the special inter-dependencies that are developed by this integration. Here the market of maritime facilities is modelled as an n-person non-cooperative game among port authorities who control the attractiveness of their terminal facilities. By taking the above interdependencies into consideration, optimal decisions of port authorities are obtained by extending the classical single leader-multiple followers Stackelberg game-theoretic formulation of the Network Design Problem (NDP) to its complete form of multiple leaders-multiple followers Competitive NDP (CNDP). The estimation of the equilibrium point of the above formulation is made by incorporating a novel evolutionary game-theoretic genetic operator into a hybrid Genetic Algorithm. The results from the application of the proposed framework into a realistic part of the East Mediterranean freight network show the potential of the method to support decisions of port authorities concerning future infrastructure investments.

Keywords: Freight Networks, Maritime Container Terminal Systems, Competitive Network Design, Co-Evolutionary non-Cooperative Algorithms.

1 Introduction

By taking into consideration that freight systems (especially those of the containerized cargo) are forming a complex network of interrelationships among operators, infrastructure managers and cargo shippers (spanning both in spatial as to temporal domain), the system of cargo operations can be viewed as a deregulated market of increased competition. In the current paper the system of combined maritime/truck-train transportation is modeled, identifying the multiple differences among the alternative modes of transport composing the supply-chain. By taking into consideration the increasing congestion phenomena on port container terminals (especially those in the major maritime markets), the Stochastic User-Equilibrium (SUE) assumption is adopted in order to predict container volumes over the network links.

Presenting a brief background on the subject of optimal design and planning of maritime container facilities, a large body of literature can be identified [1,2]. In

particular the demand among competing ports has been modeled by mathematical programming techniques [3] and discrete choice models [4], [5] and [6]. A mathematical programming model for optimal allocation of demand to alternative ports controlled by a single authority has been presented in [7]. In these studies both deliberate and controlled decisions of carriers have been captured but no planning target has been incorporated. In [8] optimal planning of a system of ports has been presented, but there is a single system designer who is making decisions for the whole system. In the current formulation, this problem has been extended to the more complex case where multiple competing terminals are engaged in a situation where there are aiming to the simultaneous optimization of development strategies for future infrastructure expansions, by considering that the market of cargo services compose a complex net of interrelated and conflicting interests among its participants.

The operation of the cargo system movements in alternative routes, ports and means, is composed here by following the non-supervised (free choice) model where shippers are flexible to choose routes and transport means. As has been already identified in [6], carriers' strategies primarily involve long-run decisions of deploying vessels to routes and short-term decisions of assigning shipments to vessels.

By recognizing that carriers can use alternative routes for servicing demand, port authorities have intensified the competition for attracting cargo, by improving, primarily, service time and pricing. In turn, carriers are making decisions by taking into account the cost of each alternative route available for every Origin-Destination (OD) $r - s$ pair, which is affected by choices made by all other carriers (since congestion effects are identified). This load-responsive port performance assumption, leads to the situation where the loading of each port is estimated as the equilibrium point of carriers' route choice non-cooperative game. An optimization formulation of non-cooperative system improvement is proposed next, extending the classical scheme of the Network Design Problem (NDP) by replacing the single decision maker (network operator) with a decentralized multi-decision maker's process leading to the Competitive Network Design Problem (CNDP). In order to address the formulation of the current setup, a decentralized competitive Intelligent Agent platform is constructed, while computational experience of the framework application to a realistic port system of East Mediterranean is followed. At the final part, conclusions and outlook of the proposed method are drawn.

2 Formulation of the Optimal Competitive Port System Design Problem

The case of designing future expansions to the infrastructure of maritime container terminals is modeled here as non-cooperative multi-level game in cascade. Terminals are competing for loads by controlling their operating conditions (their capacity choices in terms of practical throughput) while cargo shippers/carriers are also making non-cooperatively mode and route choices aiming to minimize total service costs. The case where multiple designers are competitively aiming to optimize their objectives has not received much attention in the literature. The properties of the problem (existence, uniqueness etc) for simplified conditions have been presented in [9] while some insightful notes have been offered in [10], [11] and [12]. Here the formulation

of the CNDP will follow that of the multi-level programming problems. Thus, taking that the set of alternative port terminals $i \in P$ over a study area are competing for profits Q_i , that profits are related only to container volumes serviced by each terminal x_i (expressed here in Twenty-foot Equivalent Units – TEUs) and investments b_i are related only to infrastructure improvement w_i , then the formulation of the problem can be stated as:

Upper-Level:

$$\max_w Z = [Q_1(w, x), Q_2(w, x), \dots, Q_i(w, x)], \quad \forall i \in P \quad (1)$$

subject to:

$$0 \leq w_i \leq \max w_i \quad \forall i \in P \quad (2)$$

$$0 \leq b_i \leq \max b_i \quad \forall i \in P \quad (3)$$

Lower-Level:

$$\min_x G(x) = - \sum_{rs} q_{rs} E \left[\min_{k \in K_{rs}} \{c_{rs}^k\} | C^{rs}(x) \right] + \sum_a x_a c_a(x_a) - \sum_a \int_0^{x_a} c_a(\omega) d\omega \quad (4)$$

subject to:

$$f_k^{rs} = P_k^{rs} q_{rs}, \quad \forall k, r, s \quad (5)$$

$$x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{ak}^{rs}, \quad \forall a \in A \quad (6)$$

$$f_k^{rs}, x_a \geq 0, \quad \forall a \in A \quad (7)$$

where Z is a vector objective function containing the performance functions whose elements are simultaneously maximized. Profits here are calculated as:

$$Q_i = CC_i x_i - \lambda_i w_i \quad (8)$$

where CC_i stands for marginal handling charges, x_i for TEUs volume and λ_i for the marginal cost of w_i improvement (in TEUs throughput) for each port container terminal i . Also, it is straightforward to derive that $b_i = \lambda_i w_i$ and that equations (2) and (3) stands for the physical and budgetary constraints respectively.

The estimation of the volume of TEUs that will be serviced by terminal $i \in P$, are estimated by a network model able to provide the equilibrium flows on links $a \in A$, where some of them were taken to be container port terminals ($P \subseteq A$). Here the estimation of the equilibrium flows is based on the Stochastic User Equilibrium (SUE) assumption where the error perception of users on route choices among alternative paths k connecting O-D pairs $r - s$ is taken to follow a Gumbel distribution, leading to a logit-based loading process [13]. Following this assumption, link flows are estimated in the lower-level minimization problem (equation (4)), where the

loading to each feasible path k connecting $r - s$ O-D pairs, f_k^{rs} , is probabilistically (p_k^{rs}) estimated based on route cost C^{rs} , while in constraint (5) the total assigned flows in alternative paths is required to equal total demand q_{rs} . Also, as stated in constraint (6), the flow on each link, x_a , is taken to be the sum of all path flows utilizing link a as obliged by the network incidence matrix operator δ_{ak}^{rs} , while inequalities in constraint (7), ensures the non-negativity of path and link flows. In order to address the above non-cooperative non-linear and non-convex formulation of the CNDP, a decentralized hybrid optimization platform based on genetic evolution is proposed next.

3 Co-evolutionary Algorithms for Addressing Competitive Network Design Problems

The current study proposes a decentralized optimization structure in order to address the problem of the CNDP. The proposed method is based on the structure of Agent-Based Simulation (ABS), which has been used to address similar complex decentralized optimization problems. Applications of the ABS framework can be identified in advanced control strategies, complex biological systems and population dynamics [14]. At the current problem setup, each intelligent agent (container terminal port authority) attempts to optimize its performance (profits maximization) in relation to the behavior of all others agents. The present evolutionary game theoretic setup utilizes concepts of population dynamics and Evolutionary Stable Strategies (ESSs), as they were first introduced by [15]. In order to estimate the equilibrium solution of the earlier described game-theoretic formulation of the competitive maritime container terminal design problem, optimal strategy of the intelligent agents is obtained by forming a platform of mutual genetic evolution among them, able to lead to ESS. In particular, the evolution of the agents' strategies should be made by taking into account two perspectives:

- Evolve to optimal strategies for each agent, while
- Adapting to the strategies of other agents

Here a novel evolutionary tactic is introduced for estimating the strategy of each agent, based on a hybridized version of Genetic Algorithms (GAs) [16]. GAs are population-based stochastic approximation optimization methods on the basis of Darwinian laws of natural evolution. Omitting (due to space limitations) the description of standard GAs that can be found in specialized textbooks, only the co-evolutionary properties of the proposed algorithm will be provided here.

The present co-evolutionary algorithm is suitably hybridized in order to model the interaction among competitive agents, based on the mechanics of mutual evolution of all agents participating in the game. In particular, the evolution of solutions (individuals) is performed by establishing a ‘communication link’ among the alternative agents leading to a co-evolutionary process. Thus, the evolution of the population of each agent is made by allowing them to monitor the evolution of strategies of all other agents, and adjust their strategies with respect to theirs. By assuming that each and

every agent is non-cooperatively aiming the optimization of its objective function, the algorithm's convergence state stands for an ESS of the co-evolution and thus an equilibrium of the game among them [15]. The co-evolution of the agents strategies is introduced here using a modified GA in the performance evaluation phase. This phase differs from that of the standard GA, since each candidate solution is tested against a random sample of the other agent strategies, in order to investigate its average performance with respect to possible responses of the other agents. The idea of establishing a communication link in evolutionary algorithms has been presented by [17] in an attempt to parallelize a computational procedure in order to improve convergence time. Similar applications have been presented in the recent years [18], [19]. In those applications, no competing strategies have been added among the various parts of the evolving system. Although it will be demonstrated in the next sections, it is already evident that the computational effort of the co-evolutionary algorithm is significantly increased compared to the standard GA.

4 Computational Experience

4.1 Network Formation

The application of the above described formulation of the CNDP is made here on a directed graph representing a sub-network composed of 3 significant maritime container terminals, namely Gioia Tauro (in Italy), Piraeus and Thessaloniki (in Greece), that are competing for the provision of services for flows originated/destinated from/to Asia, Western Europe and the countries of the Black Sea (Figure 2). The total service network is divided into four subsets namely, truck links L , train links T , nautical links N and port links P . In the current analysis a demand matrix of $20 \times 20 = 400$ OD pairs is considered, while no transit cargo has been taken into account.

Starting from the link cost function, total cost is provided by the following function:

$$c_a(x) = TC_a l_a + cc_a(x_a) \quad (9)$$

where the first component is related to the total path cost as a function of links fixed costs (handling and transportation costs) and the second component introduces congestion (or delay) costs at each link. In particular, in road, train and nautical links the cost coefficient, TC_a , is related only to the link length, l_a in the following form:

$$TC_a = l_a CpK_a, \quad \forall a \in A \cap P \quad (10)$$

where distance l_a is multiplied by a Cost-per-Kilometer (CpK) rate, differentiated for train, truck and nautical links. In the port links ($a \in P$), $TC_a \equiv CC_i$ since fixed costs are corresponding to handling charges as described by equation (8). The nominal values used of the CpK for the alternative transportation mean links are $CpK_a = 1$ €/Km, $\forall a \in L$ (Truck Links), $CpK_a = 0.9$ €/Km, $\forall a \in T$ (Train Links) and $CpK_a = 0.1$ €/Km, $\forall a \in N$ (Nautical Links) respectively.

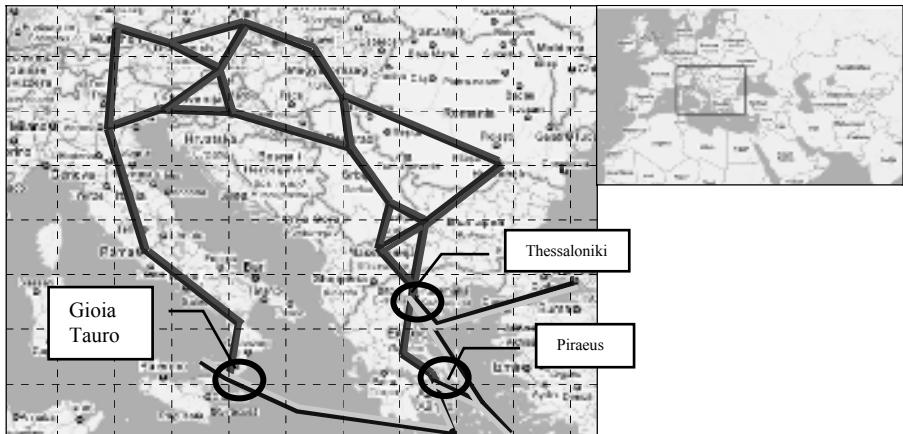


Fig. 1. The coding of the study area's network

It is noted that train links are considered as more attractive, since they are 'cheaper', nevertheless it should be recalled that these values are appropriately scaled in order to reflect the total transportation costs (that include attributes such as the value of travel time, handling costs, insurance, etc).

Congestion costs follow the concepts introduced in [20] and are only estimated in ports, since congestion effects for the particular analysis scale in interurban truck or train freight movements are not considered here. In order to estimate congestion effects (mainly corresponding to handling delays, warehouse pricing or delays in customs) on port links, a piecewise-defined function is adopted:

$$cc(x_a) = \begin{cases} c_0 \exp\left(\frac{x_a}{Y_a}\right), & \forall a \in P \\ 0, & \forall a \notin P \end{cases} \quad (11a)$$

$$(11b)$$

where c_0 stands for the base cost related to the case where port a serve no flow (here $c_0 = 200 \text{ €/TEU}$) and Y_a for the nominal maximum throughput in TEUs, calculated as $Y_a = Y_a^0 + w_a$, $\forall a \in P$, where Y_a^0 stands for the existing throughput of port a (in TEUs) and w_a for its improvement. The estimation of the necessary funds for providing additional infrastructure is estimated following the assumption that the increment of the practical throughput by 1 million TEUs/year costs 100 million €. Finally, the total available budget for each port i is taken equal to 150 million €. The total physical availability for capacity improvement is taken to be 1.5 million TEUs/year. Also, the existing practical throughput Y_a^0 of the three ports are 4 million TEUs/year, 2 million TEUs/year and 700 thousand TEUs/year for container terminals of Gioia Tauro, Piraeus and Thessaloniki respectively. Finally, the CC_i handling charges are

taken as flat equal to 100€/TEU representing the marginal cost pricing for providing cargo handling services (containing loading, unloading and land handling services). All costs, although indicative, reflect realistic conditions in Mediterranean ports.

Since the network model aims to capture the conditions over a large, multi-modal, location-specific and non-homogeneous area, a multi-dimensional discrete choice model was used (rather a multinomial logit model) for modeling port selection (as part of the route choice procedure). Thus, the probability P_k^{rs} , is estimated based on a model able to identify two stages in the route choice process, namely the choice of a port and then the choice of a transport mode (and thus route). In order to use such a model, total path utility U_k is structured in the following form:

$$U_k^{rs} = V_k^{rs}(a) + V_k^{rs}(i) + \varepsilon_{k,ia}^{rs} = \theta \sum_{a \in A} \delta_{a,k}^{rs} c_a(x_a) + \sum_{i \in C} \beta_i \delta_{i,k}^{rs} + \varepsilon_{k,pm}^{rs} \quad (12)$$

where θ is the dispersion parameter of the route cost perception (here common for all modes), $\delta_{i,k}^{rs}$ is the binary variable connecting route k and port i (see section 2) while β_i stands for a classification parameter of port preferences. Then the probability of choosing path k , is estimated as:

$$p_k^{rs}(i, a) = e^{U_k^{rs}} / \sum_{k \in K} e^{U_k^{rs}} \quad (13)$$

where K^{rs} stands for the paths set connecting $r - s$ OD pair. By using the above Joint Logit Model, the SUE network flows are estimated by the method of successive averages (MSA) algorithm, where 50 iterations (executed in almost 1 sec of CPU time on a quad-core PC) are sufficient for providing equilibrium flows.

4.2 Computational Experiments

For gaining some insights of the character and nature of problems concerning network development as earlier stated, some computational experiments have been conducted. The coding has been done in a well known commercial computational platform able to perform parallel and distributed computing, while the computation facility was a quad CPU PC (4x2.4 GHz with 8 GB of RAM). The configuration of the co-evolutionary algorithm used here utilizes a population for each agent of 50 individuals, while two stopping criteria have been chosen: the number of generations or no significant improvement ($\leq 2\%$) for 50 consecutive generations. These criteria provide enough opportunity for evolution within reasonable computational effort. Initially, a rather limited size of 10% (5 individuals per agent) random sample has been selected for testing the performance of each agent's individual against others agents, leading to 3(agents)×50(individuals)×5(random sample from the first agent)×5 (random sample from the second agent)=3750 evaluations per generation and a maximum number of 1000 generations is adopted as stopping criterion executed in almost 96 hours. This configuration provides an evolutionary process that in the early stages of evolution achieves to amend the performance of the port terminals, but as it can be

observed (Figure 2a) eventually the evolution oscillates around an attractor point, without being able to provide an ESS.

Then the random sample is increased to 30% of population (15 individuals per agent) leading to $3 \times 50 \times 15 \times 15 = 33750$ evaluations per generation but this time 200 is the maximum number of generations before stopping executed in almost 360 hours. Within this setup the evolutionary process was able to provide a reasonable range of solutions that can be accepted as ESS, as indicating the convergence graph of Figure 2b.

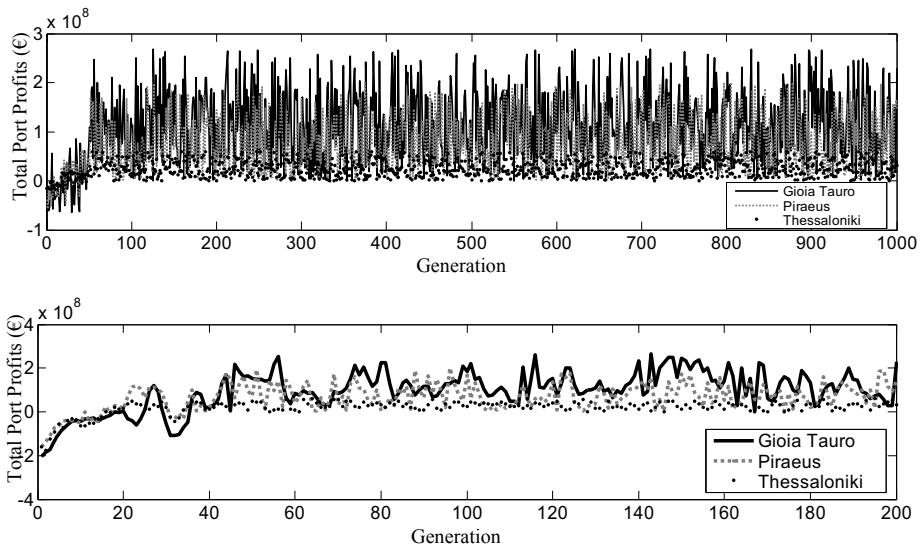


Fig. 2. The process of genetic co-evolution of the multi-agent system with 10% (a) and 30% (b) co-evolution sample respectively

The results of the algorithm is taken to be the strategy offered by the best individual of each agent population at the final generation. This is so since it must be recalled that, co-evolution has two dimensions: (a) evolution with respect to the opportunities of each agent and (b) adaptation to the evolution of the competing agents, meaning that the best strategy of earlier stages of evolution can be addressed by responses of the other agents such as not to be optimal in later stages. It should be also mentioned here that the co-evolution strategy selected actually aims to coordinate solutions towards an equilibrium point (ideally a Pareto-efficient one). Nevertheless since in non-cooperative game-theoretic situations (such as the present one) multiple equilibriums can exist, equilibrium selection mechanisms can be implemented within the proposed framework.

As far as the results of the CNDP are concerned, it is observed that the optimal strategy for the port of Piraeus is to invest more than the other two competitors, aiming to improve its market position, suggesting that its location is providing a competitive advantage in attracting service flows, since there is a significant increase in service demand.

Table 1. Results of design strategies for the competing container terminals

	Gioia Tauro (I)	Piraeus (Gr)	Thessaloniki (Gr)
Port Profits (€)	229.685.410	143.741.376	30.718.891
Port Investment Strategy (TEUs)	187.452	248.237	77.853
Port Service Demand (TEUs)	2.484.306 (-5%)	1.685.651(+11%)	385.042(-12%)
Congestion Costs/TEU (€)	361,98 (-6%)	423,30 (-0%)	328,10 (-10%)

Note: Values in parenthesis show differences to the no-investment scenario.

5 Conclusions

In the current paper, the design process of future development of a system of container port terminals has been analyzed by identifying interactions among demand and supply for the provision of handling services. The formulation of alternative design tactics extend classical game theoretic schemes adopted in network design problems, as that of the Stackelberg single leader–multiple followers game of complete information, to the case where multiple authorities are engaged in the design of distinct parts of the network. The solution of this case is taken as an equilibrium point of a non-cooperative games sequence of complete information. The complete form of the game is applied to a realistic part of the containerized cargo network of eastern Mediterranean, which forms a market of intense competition among cargo shippers and ports. Results of alternative co-evolutionary scenarios have provided interesting comparisons giving useful insights in such processes.

The proposed framework provided encouraging results in deploying such advanced evolutionary schemes for addressing complex optimization cases as those of the competitive version of the NDP. Also, particular behavioral aspects of market operations like competition, cooperation and coordination can be modeled through the proposed framework, giving rise its utilization in designing and implementation instances. Although there is potential of employing the proposed evolutionary framework in such circumstances, more elaboration should be done in investigating the performance of alternative co-evolutionary schemes.

References

1. Gunther, H.-O., Kim, K.H. (eds.): Container Terminals and Automated Transport Systems. Springer, Heidelberg (2005)
2. Wang, T.-F., Cullinane, K., Song, D.-W.: Container port production and economic efficiency. Palgrave MacMillan, NY (2005)
3. Perakis, A.N., Jaramillo, D.I.: Fleet deployment optimization for liner shipping: Part 1. Background, problem formulation, and solution approaches. Maritime Policy & Management 18(3), 183–200 (1991)
4. Garrido, A.R., Mahmassani, S.H.: Forecasting freight transportation demand with the space-time multinomial probit model. Transportation Research Part B 34, 403–418 (2000)

5. Veldman, J.S., Bückmann, H.E.: A Model on container port competition: An application for the west European container hub-ports. *Econ. & Logistics* 5, 3–22 (2003)
6. Malchow, B.M., Kanafani, A.: A disaggregate analysis of port selection. *Transportation Research Part E* 40, 317–337 (2004)
7. Zografos, K., Martinez, W.: Improving the performance of a port system through service demand reallocation. *Transportation Research B* 24B(2), 79–97 (1990)
8. Musso, E., Ferrari, C., Benacchio, M.: On the global optimum size of port terminals. *International Journal of Transport Economics* 26(3), 415–437 (1999)
9. Lederer, J.P.: A competitive network design problem with pricing. *Transportation Science* 27(1), 25–38 (1993)
10. De Palma, A., Thisse, J.-F.: Competition on Networks: Some Introductory Notes. *Transportation Science* 27(1), 1–3 (1993)
11. Fisk, C.S.: Spatial price equilibrium on congested networks. *Transportation Research B* 21(3), 175–182 (1987)
12. Arrow, K.J., Debreu, G.: Existence of an Equilibrium for a Competitive Economy. *Econometrica* 22, 265–290 (1954)
13. Sheffi, Y.: *Urban Transportation Networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall, Englewood Cliffs (1985)
14. Weiss, G. (ed.): *Multiagent Systems. A modern approach to distributed intelligence*. MIT Press, Cambridge (1999)
15. Maynard Smith, J.: *Evolution and the Theory of Games*. Cambridge University Press, Cambridge (1982)
16. Holland, J.H.: *Adaptation in Natural and Artificial Systems*, 2nd edn. MIT Press, Cambridge (1975)
17. Hillis, W.D.: Coevolving parasites improve simulated evolution as an optimization procedure. *Physica D* 42, 228–234 (1990)
18. Rosin, C.D., Belew, R.K.: New methods for competitive co-evolution. *Evol. Comput.* 5(1), 1–29 (1997)
19. De Jong, E.D., Pollack, J.B.: Ideal evaluation from co-evolution. *Evol. Comput.* 12(2), 159–192 (2004)
20. Daganzo, F.C.: Crane productivity and ship delay in ports. *Transportation Research Record* 1251, 1–9 (1989)

Author Index

- Alba, Enrique 11, 755
Alfaro-Cid, Eva 635
Almodóvar, Máximo 808
Alonso-Garrido, Oscar 132
Amari, Noriyuki 273
Amodeo, Lionel 420, 798
Amoretti, Michele 61
Araki, Kenji 452
Arias, Daniel 755
Audouard, Pierre 323
Aydin, Mehmet E. 71
Azzini, Antonia 99, 213
- Bandyopadhyay, Sanghamitra 426
Bánhelyi, Balázs 87
Banzhaf, Wolfgang 172
Biazzini, Marco 87
Bilotta, Eleonora 585
Bown, Oliver 528
Brabazon, Anthony 182, 233
Bradley, Robert 233
Braun, Torsten 81
Browne, Cameron 283
Browne, Tim Murray 538
Bullock, Stephen H. 375
Burschka, Stefan 105
- Cagnoni, Stefano 369
Caponio, Andrea 715
Caserta, Marco 788
Ceravolo, Francesco 123
Chan, Ching King 243
Chapman, Daniel S. 152
Chaslot, Guillaume 323
Collard, Philippe 645
Colton, Simon 283, 467
Corne, David W. 695
Corut Ergin, Fatma 31
Costa, Ernesto 263, 705
- da Costa Pereira, Célia 213
Dahlstedt, Palle 478
Dang, Jing 182
De Falco, Ivanoe 41
De Felice, Matteo 99, 123
- Della Cioppa, Antonio 41
de Moura Oliveira, Paulo B. 343
de Paly, Michael 142
De Prisco, Roberto 567
Di Caro, Gianni A. 21
Dimitriou, Loukas 818
Dorin, Alan 488, 508
Dreżewski, Rafał 223
Ducatelle, Frederick 21
Dumitrescu, D. 253
Dybala, Paweł 452
- Edelman, David 182
Eigenfeldt, Arne 498
Eldridge, Alice 508
Esparcia-Alcázar, Anna I. 635
Estevez-Tapiador, Juan M. 93
- Farooq, Muddassar 369
Femenia-Ferrer, Beatriu 635
Fonseca Ferreira, Nuno M. 343
Fox, Charles 538
- Gambardella, Luca M. 21
García, Ricardo 808
Geihs, Kurt 768
González-Botello, Miguel A. 375
Gorlitz, Christian 768
Grace, Kazjon 591
Grant, Jane 609
Greenfield, Gary 518
- Heywood, Malcolm I. 105, 603
Hinojosa-Corona, Alejandro 375
Hochreiter, Ronald 193
Hodgson, Tim 609
Hoock, Jean-Baptiste 323
Hu, Xiao-Min 51
Hubacek, Klaus 152
Hushlak, Gerald 432
- Jacob, Christian 293, 432, 442
Jaśkowski, Wojciech 333
Jelasity, Márk 87
Ji, Haru (Hyunkyoung) 597
Jin, Nanlin 152

- Kayacık, Hilmi G. 105
Keselj, Vlado 603
Khemka, Namrata 432
Korošec, Peter 675
Krawiec, Krzysztof 333
Kudelski, Michał 111
Kwan, Raymond 71

Leskinen, Jyri 615
Leung, Cyril 71
Lichocki, Paweł 333
Lin, Mu-Hua 462
Lipinski, Piotr 203
Liu, Lili 725
López-Espí, Pablo 132
Louchet, Jean 385
Lozano, Leonardo 573
Lung, Rodica Ioana 253
Luque, Gabriel 755

Machado, José A. Tenreiro 343
Machado, Penousal 557
Marcial-Romero, J. Raymundo 353
Macías, Demetrio 420
Maistro, Domenico 41
Maringer, Dietmar 162
Matthias, John 609
Maulik, Ujjwal 426
McCormack, Jon 528
McDermott, James 579
Medaglia, Andrés L. 573
Mehmood, Shahid 369
Meloni, Sandro 99
Mendes, Luís 343
Menezes, Telmo 263
Merelo, J.J. 635
Mihoc, Tudor Dan 253
Miller, Julian F. 405
Miranda, Eduardo 609
Molina, Guillermo 11
Moniz, Ryan D. 442
Montresor, Alberto 87
Mordonini, Monica 369
Moya, Pilar 635

Nagao, Tomoharu 395
Nakayama, Shiro 395
Neittaanmäki, Pekka 615
Neri, Ferrante 303, 615, 715
Ngueveu, Sandra Ulrich 778
Nguyen, Trung Thanh 735

O'Neill, Michael 182, 233, 579
Obrocki, Krystian 223
Olague, Gustavo 375, 414
Orfila, Agustín 93
Ortiz-García, Emilio G. 132

Pacut, Andrzej 111
Pantano, Pietro 585
Parreño, Francisco 808
Pérez-Bellido, Á.M. 132
Perez, Cynthia B. 414
Perez, Julien 323
Phon-Amnuaisuk, Somnuk 547, 625
Pizzuti, Stefano 123
Podlich, Alexander 768
Portilla-Figueras, Antonio 132
Pošík, Petr 685
Preuss, Mike 665
Prins, Christian 778, 798
Prodhon, Caroline 420
Ptaszynski, Michał 452
Puente, Cesar 375

Reddin, John 579
Reinhard, Kai 768
Ribagorda, Arturo 93
Rimmel, Arpad 323
Rizzuti, Costantino 585
Rohlfshagen, Philipp 745
Romero, Juan 557
Rosário, Maria J. 343
Rzepka, Rafal 452

Saha, Indrajit 426
Saks, Philip 162
Salcedo-Sanz, Sancho 132
Sánchez, David Ricardo 798
Santos, Antonino 557
Sapin, Emmanuel 385
Saunders, Rob 591
Scafuri, Umberto 41
Seredynski, Franciszek 1
Sharman, Ken 635
Sheri, Guleng 695
Shirakawa, Shinichi 395
Šilc, Jurij 675
Simões, Anabela 705
Siwik, Leszek 223
Skaruz, Jarosław 1

- Slowik, Adam 363
Smith, Stephen L. 405
Smith, Stephen V. 375
Solteiro Pires, Eduardo J. 343
Stathopoulos, Antony 818
Szeto, Kwok Yip 243
- Tarantino, Ernesto 41
Tettamanzi, Andrea G.B. 99, 213
Teytaud, Fabien 655
Teytaud, Olivier 323, 655
Tirronen, Ville 303
Tomassini, Marco 645
Tominaga, Kazuto 273
Torres, Pedro 467
- Uyar, Şima 31, 762
Uyar, H. Turgut 762
- Vanneschi, Leonardo 645
Vaz, João Caldinhas 343
Velasco, Nubia 573
Venegas, Héctor A. Montes 353
Verel, Sébastien 645
Vial, Alexandre 420
Voß, Stefan 788
Völk, Katharina 405
von Mammen, Sebastian 293
- Wakefield, Graham 597
Wälchli, Markus 81
Wang, Dingwei 725
Weber, Matthieu 303
Weicker, Karsten 313
Weicker, Nicole 313
Weise, Thomas 768
Wilson, Garnett 172
Wolfler-Calvo, Roberto 778
Wolkowicz, Jacek 603
Wozabal, David 193
- Yalaoui, Farouk 420
Yang, Hsiao-Fang 462
Yang, Shengxiang 725
Yao, Xin 735, 745
Yayımlı, Ayşegül 31
- Zaccagnino, Rocco 567
Zell, Andreas 142
Zhan, Zhi-hui 117
Zhang, Jie 71
Zhang, Jun 51, 117
Zhang, Li-Ming 51
Zincir-Heywood, A. Nur 105
Zurada, Jacek M. 363