LNCS 6625

Cecilia Di Chio et al. (Eds.)

Applications of Evolutionary Computation

EvoApplications 2011: EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM, and EvoTRANSLOG
Torino, Italy, April 2011, Proceedings, Part II

2 Part II

Springer

Lecture Notes in Computer Science 6625

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
TU Dortmund University, Germany
Madhu Sudan
Microsoft Research, Cambridge, MA, USA
Demetri Terzopoulos
University of California, Los Angeles, CA, USA
Doug Tygar
University of California, Berkeley, CA, USA
Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Cecilia Di Chio   Anthony Brabazon
Gianni A. Di Caro   Rolf Drechsler
Muddassar Farooq   Jörn Grahl   Gary Greenfield
Christian Prins   Juan Romero
Giovanni Squillero   Ernesto Tarantino
Andrea G.B. Tettamanzi   Neil Urquhart
A. Şima Uyar (Eds.)

Applications of Evolutionary Computation

EvoApplications 2011: EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM, and EvoTRANSLOG
Torino, Italy, April 27-29, 2011
Proceedings, Part II

Springer

Volume Editors

see next page

Cover illustration:
"Globosphere" by Miguel Nicolau and Dan Costelloe (2010),
University of Dublin, Ireland

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-20519-4 e-ISBN 978-3-642-20520-0
DOI 10.1007/978-3-642-20520-0
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011925061

CR Subject Classification (1998): F.1, D.2, C.2, I.4, I.2.6, J.5

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

© Springer-Verlag Berlin Heidelberg 2011
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.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Volume Editors

Cecilia Di Chio
cdichio@gmail.com

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

Gianni A. Di Caro
"Dalle Molle" Institute for
Artificial Intelligence (IDSIA)
Lugano, Switzerland
gianni@idsia.ch

Rolf Drechsler
Universität Bremen, Germany
drechsle@informatik.uni-bremen.de

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

Jörn Grahl
Department of Information Systems
Johannes Gutenberg University,
Germany
grahl@uni-mainz.de

Gary Greenfield
University of Richmond, USA
ggreenfi@richmond.edu

Christian Prins
Technical University of Troyes, France
christian.prins@utt.fr

Juan Romero
Facultad de Informatica
University of A Coruña, Spain
jj@udc.es

Giovanni Squillero
Politecnico di Torino, Italy
giovanni.squillero@polito.it

Ernesto Tarantino
Institute for High Performance
Computing and Networking, Italy
ernesto.tarantino@na.icar.cnr.it

Andrea G. B. Tettamanzi
Università degli Studi di Milano, Italy
andrea.tettamanzi@unimi.it

Neil Urquhart
Centre for Emergent Computing
Edinburgh Napier University, UK
n.urquhart@napier.ac.uk

A. Şima Uyar
Dept. of Computer Engineering
Istanbul Technical University, Turkey
etaner@cs.itu.edu.tr

Preface

The EvoApplications conference brings together many researchers working in all aspects of Evolutionary Computation. Evolutionary Computation is based on the essential operators of natural evolution, i.e., reproduction, variation and selection. Researchers working in the field of Evolutionary Computation use these operators to solve all kinds of problems in optimization, machine learning and pattern recognition. The present volume presents an overview of the latest research in Evolutionary Computation. Areas where evolutionary computation techniques have been applied range from telecommunication networks to complex systems, finance and economics, games, image analysis, evolutionary music and art, parameter optimization, scheduling and logistics. These papers may provide guidelines to help new researchers tackling their own problem using Evolutionary Computation.

The current volume represents roughly half of the papers accepted by EvoApplications 2011. The conference EvoApplications has been in existence since 2010 but actually originated from EvoWorkshops in 1998. Thus, for over 13 years, this event has brought together researchers from all around the world for an exchange of ideas. The EvoApplications conference itself adapts to the need of the participating researchers, with old events disappearing and new events appearing covering hot research topics. Some events have matured into conferences such as EuroGP in 2000, EvoCOP in 2004, and EvoBIO in 2007.

EvoApplications is part of EVO*, Europe's premier co-located events in the field of evolutionary computing (EC). EVO* was held from the 27th to the 29th of April 2011 in the beautiful city of Torino (Italy), which, having been the first capital city of Italy, held major celebrations for the 150th anniversary of national unity. Evo* 2011 included, in addition to EvoApplications, EuroGP, the main European event dedicated to genetic programming; EvoCOP, the main European conference on evolutionary computation in combinatorial optimization; EvoBIO, the main European conference on EC and related techniques in bioinformatics and computational biology. The proceedings for all of these events, EuroGP 2011, EvoCOP 2011 and EvoBIO 2011, are also available in the LNCS series (volumes 6621, 6622, and 6623, respectively).

Moreover, thanks to the large number of submissions received, the proceedings for EvoApplications 2011 are divided into two volumes. The present volume, which contains contributions for EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM and EvoTRANSLOG; and volume one (LNCS 6624), which contains contributions for EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM and EvoSTOC.

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 to simply 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.

EvoApplications 2011 consisted of the following individual events:

- *EvoCOMNET*, the 8th European Event on the Application of Nature-Inspired Techniques for Telecommunication Networks and Other Parallel and Distributed Systems
- *EvoCOMPLEX*, the 2nd European Event on Evolutionary Algorithms and Complex Systems
- *EvoFIN*, the 5th European Event on Evolutionary and Natural Computation in Finance and Economics
- *EvoGAMES*, the 3rd European Event on Bio-inspired Algorithms in Games
- *EvoHOT*, the 6th European Event on Bio-inspired Heuristics for Design Automation
- *EvoIASP*, the 13th European Event on Evolutionary Computation in Image Analysis and Signal Processing
- *EvoINTELLIGENCE*, the 2nd European Event on Nature-Inspired Methods for Intelligent Systems
- *EvoMUSART*, the 9th European Event on Evolutionary and Biologically Inspired Music, Sound, Art and Design
- *EvoNUM*, the 4th European Event on Bio-inspired Algorithms for Continuous Parameter Optimization
- *EvoSTIM*, the 6th European Event on Scheduling and Timetabling
- *EvoSTOC*, the 8th European Event on Evolutionary Algorithms in Stochastic and Dynamic Environments
- *EvoTRANSLOG*, the 5th European Event on Evolutionary Computation in Transportation and Logistics

EvoCOMNET addresses the application of EC techniques to problems in distributed and connected systems such as telecommunication and computer networks, distribution and logistic networks, interpersonal and inter-organizational networks, etc. To address the challenges of these systems, this event 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.

EvoCOMPLEX covers all aspects of the interaction of evolutionary algorithms (and metaheuristics in general) with complex systems. Complex systems are ubiquitous in physics, economics, sociology, biology, computer science and many other scientific areas. Typically, a complex system is composed of smaller aggregated components, whose interaction and interconnectedness are non-trivial. This leads to emergent properties of the system, not anticipated by its isolated components. Furthermore, when the system behavior is studied from a temporal perspective, self-organization patterns typically arise.

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 all bio-inspired heuristics applied to the electronic design automation. The event's goal is to show the latest developments, industrial experiences and successful attempts to *evolve rather than* design new solutions. EvoHOT 2011 allowed one both to peek into the problems that will be faced in the next generation of electronics, and to demonstrate innovative solutions to classic CAD problems, such as fault tolerance and test.

EvoIASP, the longest-running of all EvoApplications which celebrated its thirteenth 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.

EvoINTELLIGENCE is devoted to the use of nature-inspired methods to create intelligent systems. The scope of the event includes research in evolutionary robotics, artificial life and related areas. EvoIntelligence research also includes research in creating intelligent behavior that can be found in everyday devices such as a digital video recorder or smart phone.

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 event 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 development and collaboration among researchers.

EvoNUM aims at applications of bio-inspired algorithms, and cross-fertilization between these and more classic numerical optimization algorithms, to continuous optimization problems in engineering. It deals with theoretical aspects and 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.

EvoSTIM presents an opportunity for EC researchers in the inter-related areas of planning, scheduling and timetabling to come together, present their latest research and discuss current developments and applications.

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 event that provided a platform to present and discuss the latest research in this field.

EvoTRANSLOG deals with all aspects of the use of evolutionary computation, 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 event aims at design and optimization techniques such as EC approaches allowing the use of computer systems for systematic design, optimization and improvement of systems in the transportation and logistics domain.

Continuing in the tradition of adapting the list of the events to the needs and demands of the researchers working in the field of EC, two events were resumed this year: EvoHOT, the 6th European Event on Bio-inspired Heuristics for Design Automation, and EvoSTIM, the 6th European event on Scheduling and Timetabling.

The number of submissions to EvoApplications 2011 was again high, cumulating 162 entries (with respect to 143 in 2009 and 191 in 2010). The following table shows relevant statistics for EvoApplications 2011, where the statistics for the 2010 edition are also reported:

Event	2011			Previous edition		
	Submissions	Accept	Ratio	Submissions	Accept	Ratio
EvoCOMNET	15	8	53%	17	12	71%
EvoCOMPLEX	11	5	45%	12	6	50%
EvoENVIRONMENT	-	-	-	5	4	80%
EvoFIN	8	6	75%	17	10	59%
EvoGAMES	17	11	65%	25	15	60%
EvoHOT	7	5	71%	-	-	-
EvoIASP	19	7	37%	24	15	62%
EvoINTELLIGENCE	5	3	60%	8	5	62%
EvoMUSART	43	24	56%	36	16	44%
EvoNUM	9	5	56%	25	15	60%
EvoSTIM	9	4	44%	-	-	-
EvoSTOC	8	5	63%	11	6	54%
EvoTRANSLOG	11	4	36%	11	5	45%
Total	162	87	54%	191	109	57%

As for previous years, accepted papers were split into oral presentations and posters. However, this year, the paper length for these two categories was the same for all the events. The low acceptance rate of 54% for EvoApplications 2011, along with the significant number of submissions, is an indicator of the high quality of the articles presented at the events, showing the liveliness of the scientific movement in the corresponding fields.

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

– The University of Torino - School for Biotechnologies and Molecular Biotechnology Center, for supporting the local organization

- The Human Genetics Foundation of Torino (HuGeF), the Museum of Human Anatomy ("Luigi Rolando") and the Museum of Criminal Anthropology ("Cesare Lombroso") for their patronage of the event
- The Centre for Emergent Computing at Edinburgh Napier University, UK, for administrative help and event coordination

We want to especially acknowledge our invited speakers: Craig Reynolds (Sony Computer Entertainment, USA) and Jean-Pierre Changeux.

Even with an excellent support and location, an event like EVO* would not have been feasible without authors submitting their work, members of the Programme 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 such a type of conference with a large number of different organizers and different opinions would be unmanageable. Further thanks to the local organizer Mario Giacobini for making the organization of such an event possible and successful. Last but surely not least, we want to especially acknowledge Penousal Machado for his hard work as Publicity Chair and webmaster (assisted by Pedro Miguel Cruz and João Bicker), and Marc Schoenauer for his continuous help in setting up and maintaining the MyReview management software.

April 2011

Cecilia Di Chio
Anthony Brabazon
Gianni Di Caro
Rolf Drechsler
Muddassar Farooq
Jörn Grahl
Gary Greenfield

Christian Prins
Juan Romero
Giovanni Squillero
Ernesto Tarantino
Andrea G. B. Tettamanzi
Neil Urquhart
A. Şima Uyar

Organization

EvoApplications 2011 was part of EVO* 2011, Europe's premier co-located events in the field of evolutionary computing, which also included the conferences EuroGP 2011, EvoCOP 2011 and EvoBIO 2011.

Organizing Committee

EvoApplications Chair	Cecilia Di Chio, UK
Local Chairs	Mario Giacobini, University of Torino, Italy
Publicity Chair	Penousal Machado, University of Coimbra, Portugal
EvoCOMNET Co-chairs	Gianni A. Di Caro, IDSIA, Switzerland Muddassar Farooq, National University of Computer and Emerging Sciences, Pakistan Ernesto Tarantino, Institute for High-Performance Computing and Networking, Italy
EvoCOMPLEX Co-chairs	Carlos Cotta, University of Malaga, Spain Juan J. Merelo, University of Granada, Spain
EvoFIN Co-chairs	Anthony Brabazon, University College Dublin, Ireland Andrea G.B. Tettamanzi, University of Milano, Italy
EvoGAMES Co-chairs	Mike Preuss, TU Dortmund University, Germany Julian Togelius, IT University of Copenhagen, Denmark Georgios N. Yannakakis, IT University of Copenhagen, Denmark
EvoHOT Co-chairs	Giovanni Squillero, Politecnico di Torino, Italy Rolf Drechsler, University of Bremen, Germany
EvoIASP Chair	Stefano Cagnoni, University of Parma, Italy
EvoINTELLIGENCE Chair	Marc Ebner, University of Tübingen, Germany

EvoMUSART Co-chairs	Gary Greenfield, University of Richmond, USA
	Juan Romero, University of A Coruña, Spain
EvoNUM Co-chairs	Anna I Esparcia-Alcázar, S2 Grupo, Spain
	Anikó Ekárt, Aston University, UK
EvoSTIM Co-chairs	A. Şima Uyar, Istanbul Technical University, Turkey
	Neil Urquhart, Edinburgh Napier University, UK
EvoSTOC Co-chairs	Ferrante Neri, University of Jyväskylä, Finland
	Hendrik Richter, HTWK Leipzig University of Applied Sciences, Germany
EvoTRANSLOG Co-chairs	Christian Prins, Technical University of Troyes, France
	Jörn Grahl, Johannes Gutenberg University, Germany

Programme Committees

EvoCOMNET Programme Committee

Özgür B. Akan	Koc University, Turkey
Enrique Alba	University of Malaga, Spain
Qing Anyong	National University of Singapore, Singapore
Payman Arabshahi	University of Washington, USA
Mehmet E. Aydin	University of Bedfordshire, UK
Iacopo Carreras	CREATE-NET, Italy
Frederick Ducatelle	IDSIA, Switzerland
Luca Gambardella	IDSIA, Switzerland
Jin-Kao Hao	University of Angers, France
Malcolm I. Heywood	Dalhousie University, Canada
Kenji Leibnitz	Osaka University, Japan
Domenico Maisto	ICAR CNR, Italy
Roberto Montemanni	IDSIA, Switzerland
Conor Ryan	University of Limerick, Ireland
Muhammad Saleem	FAST National University of Computer and Emerging Technologies, Pakistan
Chien-Chung Shen	University of Delaware, USA
Tony White	Carleton University, Canada
Lidia Yamamoto	University of Strasbourg, France
Nur Zincir-Heywood	Dalhousie University, Canada

EvoCOMPLEX Programme Committee

Antonio Córdoba	Universidad de Sevilla, Spain
Carlos Cotta	Universidad de Málaga, Spain

Jordi Delgado	Universitat Politècnica de Catalunya, Spain
Marc Ebner	University of Tübingen, Germany
José E. Gallardo	Universidad de Málaga, Spain
Carlos Gershenson	UNAM, Mexico
Anca Gog	Babes-Bolyai University, Romania
Márk Jelasity	University of Szeged, Hungary
Juan Luis Jiménez	Universidad de Granada, Spain
Juan J. Merelo	Universidad de Granada, Spain
Joshua L. Payne	University of Vermont, USA
Mike Preuss	Universität Dortmund, Germany
Katya Rodríguez-Vázquez	UNAM, Mexico
Kepa Ruiz-Mirazo	Euskal Herriko Unibertsitatea, Spain
Robert Schaefer	AGH University of Science and Technology, Poland
Marco Tomassini	Université de Lausanne, Switzerland
Alberto Tonda	Politecnico di Torino, Italy
Fernando Tricas	Universidad de Zaragoza, Spain
Leonardo Vanneschi	University of Milano-Bicocca, Italy

EvoFIN Programme Committee

Eva Alfaro-Cid	Instituto Tecnológico de Informática, Spain
Alexandros Agapitos	University College Dublin, Ireland
Antonia Azzini	Università degli Studi di Milano, Italy
Anthony Brabazon	University College Dublin, Ireland
Robert Bradley	University College Dublin, Ireland
Louis Charbonneau	Concordia University, Canada
Gregory Connor	National University of Ireland Maynooth, Ireland
Ian Dempsey	Pipeline Trading, USA
Manfred Gilli	University of Geneva and Swiss Finance Institute, Switzerland
Philip Hamill	University of Ulster, UK
Ronald Hochreiter	WU Vienna University of Economics and Business, Austria
Serafin Martinez Jaramillo	Bank of Mexico, Mexico
Youwei Li	Queen's University Belfast, UK
Piotr Lipinski	University of Wroclaw, Poland
Dietmar Maringer	University of Basel, Switzerland
Michael O'Neill	University College Dublin, Ireland
Robert Schafer	AGH University of Science and Technology, Poland
Enrico Schumann	Switzerland
Andrea Tettamanzi	Università degli Studi di Milano, Italy

Nikolaos Thomaidis	University of the Aegean, Greece
Ruppa Thulasiram	University of Manitoba, Canada
Garnett Wilson	Memorial University of Newfoundland, Canada

EvoGAMES Programme Committee

Lourdes Araujo	UNED, Spain
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Luigi Barone	University of Western Australia, Australia
Robin Baumgarten	Imperial College London, UK
Paolo Burelli	IT-Universitetet i København, Denmark
Simon Colton	Imperial College London, UK
Ernesto Costa	Universidade de Coimbra, Portugal
Carlos Cotta	Universidad de Málaga, Spain
Marc Ebner	University of Tübingen, Germany
Anikó Ekárt	Aston University, UK
Anna I Esparcia Alcázar	S2 Grupo, Spain
Antonio J Fernández Leiva	Universidad de Málaga, Spain
Francisco Fernández	Universidad de Extremadura, Spain
Edgar Galvan-Lopes	University College Dublin, Ireland
Leo Galway	University of Ulster, UK
Mario Giacobini	Università degli Studi di Torino, Italy
Johan Hagelbäck	Blekinge Tekniska Högskola, Sweden
John Hallam	University of Southern Denmark
David Hart	Fall Line Studio, USA
Erin Hastings	University of Central Florida, USA
Philip Hingston	Edith Cowan University, Australia
Stefan Johansson	Blekinge Tekniska Högskola, Sweden
Krzysztof Krawiec	Poznan University of Technology, Poland
Pier Luca Lanzi	Politecnico di Milano, Italy
Simon Lucas	University of Essex, UK
Penousal Machado	Universidade de Coimbra, Portugal
Tobias Mahlmann	IT-Universitetet i København, Denmark
Hector P. Martinez	IT-Universitetet i København, Denmark
J.J. Merelo	Universidad de Granada, Spain
Risto Miikkulainen	University of Texas at Austin, USA
Antonio Mora	Universidad de Granada, Spain
Steffen Priesterjahn	University of Paderborn, Germany
Jan Quadflieg	TU Dortmund, Germany
Moshe Sipper	Ben-Gurion University, Israel
Noor Shaker	IT-Universitetet i København, Denmark
Terry Soule	University of Idaho, USA
Christian Thurau	Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme, Germany

EvoHOT Programme Committee

Angan Das	Intel Corporation, USA
Rolf Drechsler	University of Bremen, Germany
Gregor Papa	Jozef Stefan Institute, Slovenia
Javier Perez	Universidad Pedagógica y Tecnológica de Colombia, Colombia
Marco Santambrogio	Politecnico di Milano, Italy
Alberto Tonda	Politecnico di Torino, Italy
Ernesto Sanchez	Politecnico di Torino, Italy
Giovanni Squillero	Politecnico di Torino, Italy

EvoIASP Programme Committee

Antonia Azzini	Università degli Studi di Milano, Italy
Lucia Ballerini	University of Edinburgh, UK
Leonardo Bocchi	University of Florence, Italy
Stefano Cagnoni	University of Parma, Italy
Oscar Cordon	European Center for Soft Computing, Spain
Sergio Damas	European Center for Soft Computing, Spain
Ivanoe De Falco	ICAR - CNR, Italy
Antonio Della Cioppa	University of Salerno, Italy
Laura Dipietro	MIT, USA
Marc Ebner	University of Tübingen, Germany
Francesco Fontanella	University of Cassino, Italy
Şpela Ivekoviç	University of Dundee, UK
Mario Koeppen	Kyushu Institute of Technology, Japan
Krisztof Krawiec	Poznan University of Technology, Poland
Jean Louchet	INRIA, France
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
Andy Tyrrell	University of York, UK
Leonardo Vanneschi	University of Milano-Bicocca, Italy
Mengjie Zhang	Victoria University of Wellington, New Zealand

EvoINTELLIGENCE Programme Committee

Riad Akrour	INRIA Saclay Île-de-France, France
Amit Benbassat	Ben-Gurion University, Israel
Peter Bentley	University College London, UK

Stefano Cagnoni	University of Parma, Italy
Marc Ebner	Eberhard Karls Universität Tübingen, Germany
Aniko Ekart	Aston University, UK
Ian Horswill	Northwestern University, USA
Christian Jacob	University of Calgary, Canada
Gul Muhammad Kahn	University of Engineering and Technology, Pakistan
William B. Langdon	King's College, London
Penousal Machado	University of Coimbra, Portugal
Michael O'Neill	University College Dublin, Ireland
Michael Orlov	Ben-Gurion University, Israel
Thomas Ray	University of Oklahoma, USA
Marc Schoenauer	INRIA, France
Moshe Sipper	Ben-Gurion University, Israel
Giovanni Squillero	Politecnico di Torino, Italy
Ivan Tanev	Doshisha University, Japan
Mengjie Zhang	Victoria University of Wellington, New Zealand

EvoMUSART Programme Committee

Mauro Annunziato	Plancton Art Studio, Italy
Dan Ashlock	University of Guelph, Canada
Peter Bentley	University College London, UK
Eleonora Bilotta	University of Calabria, Italy
Jon Bird	University of Sussex, UK
Tim Blackwell	Goldsmiths College, University of London, UK
Oliver Bown	Monash University, Australia
Paul Brown	University of Sussex, UK
Kevin Burns	Mitre Corporation, USA
Stefano Cagnoni	University of Parma, Italy
Amilcar Cardoso	University of Coimbra, Portugal
Vic Ciesielski	RMIT, Australia
John Collomosse	University of Surrey, UK
Simon Colton	Imperial College, UK
Palle Dahlstedt	Göteborg University, Sweden
Hans Dehlinger	Independent Artist, Germany
Alan Dorin	Monash University, Australia
Scott Draves	Independent Artist, USA
Erwin Driessens	Independent Artist, The Netherlands
Carla Farsi	University of Colorado, USA
Josè Fornari	NICS/Unicamp, Brazil
Marcelo Freitas Caetano	IRCAM, France
Philip Galanter	Texas A&M College of Architecture, USA
Pablo Gervs	Universidad Complutense de Madrid, Spain
Andrew Gildfind	Google, Inc., Australia

Gary Greenfield	University of Richmond, USA
Carlos Grilo	Instituto Politècnico de Leiria, Portugal
Amy K. Hoover	University of Central Florida, USA
Andrew Horner	University of Science and Technology, Hong Kong
Christian Jacob	University of Calgary, Canada
Colin Johnson	University of Kent, UK
Craig Kaplan	University of Waterloo, Canada
William Latham	Goldsmiths College, University of London, UK
Matthew Lewis	Ohio State University, USA
Yang Li	University of Science and Technology Beijing, China
Alain Lioret	Paris 8 University, France
Penousal Machado	University of Coimbra, Portugal
Bill Manaris	College of Charleston, USA
Ruli Manurung	University of Indonesia, Indonesia
Jon McCormack	Monash University, Australia
James McDermott	University of Limerick, Ireland
Eduardo Miranda	University of Plymouth, UK
Nicolas Monmarchè	University of Tours, France
Gary Nelson	Oberlin College, USA
Luigi Pagliarini	Pescara Electronic Artists Meeting and University of Southern Denmark, Italy
Alejandro Pazos	University of A Coruna, Spain
Somnuk Phon-Amnuaisuk	University Tunku Abdul Rahman, Malaysia
Rafael Ramirez	Pompeu Fabra University, Spain
Juan Romero	University of A Coruna, Spain
Brian Ross	Brock University, Canada
Artemis Sanchez Moroni	Renato Archer Research Center, Brazil
Antonino Santos	University of A Coruna, Spain
Benjamin Schroeder	Ohio State University, USA
Jorge Tavares	University of Coimbra, Portugal
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 Programme Committee

Eva Alfaro	Instituto Tecnológico de Informática, Spain
Anne Auger	INRIA, France
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Xavier Blasco	Universidad Politécnica de Valencia, Spain

Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Ying-ping Chen	National Chiao Tung University, Taiwan
Carlos Cotta	Universidad de Malaga, Spain
Marc Ebner	Universität Würzburg, Germany
Gusz Eiben	Vrije Universiteit Amsterdam, The Netherlands
A. Şima Uyar	Istanbul Technical University, Turkey
Francisco Fernández	Universidad de Extremadura, Spain
Nikolaus Hansen	INRIA, France
José Ignacio Hidalgo	Universidad Complutense de Madrid, Spain
Andras Joo	Aston University, UK
Bill Langdon	King's College London, UK
J.J. Merelo	Universidad de Granada, Spain
Salma Mesmoudi	INRIA, France
Christian Lorenz Müller	ETH Zurich, Switzwerland
Boris Naujoks	Log!n GmbH, Germany
Ferrante Neri	University of Jyväskylä, Finland
Gabriela Ochoa	University of Nottingham, UK
Petr Pošík	Czech Technical University, Czech Republic
Mike Preuss	University of Dortmund, Germany
Günter Rudolph	University of Dortmund, Germany
Ivo F. Sbalzarini	ETH Zurich, Switzerland
Marc Schoenauer	INRIA, France
Hans-Paul Schwefel	University of Dortmund, Germany
P.N. Suganthan	Nanyang Technological University, Singapore
Ke Tang	University of Science and Technology of China, China
Olivier Teytaud	INRIA, France
Darrell Whitley	Colorado State University, USA

EvoSTIM Programme Committee

Ben Paechter	Edinburgh Napier University, UK
Emma Hart	Edinburgh Napier University, UK
Ryhd Lewis	Cardiff Business School, UK
John Levine	Strathclyde University, UK
Sanem Sariel	Istanbul Technical University, Turkey
Rong Qu	University of Nottingham, UK
Ender Ozcan	University of Nottingham, UK
Nelishia Pillay	University of KwaZulu-Natal, South Africa
Peter Cowling	University of Bradford, UK
Sanja Petrovic	University of Nottingham, UK

EvoSTOC Programme Committee

Hussein Abbass	University of New South Wales, Australia
Dirk Arnold	Dalhousie University, Canada
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Peter Bosman	Centre for Mathematics and Computer Science, The Netherlands
Juergen Branke	University of Karlsruhe, Germany
Andrea Caponio	Technical University of Bari, Italy
Ernesto Costa	University of Coimbra, Portugal
Kalyanmoy Deb	Indian Institute of Technology Kanpur, India
Andries Engelbrecht	University of Pretoria, South Africa
Yaochu Jin	Honda Research Institute Europe, Germany
Anna V. Kononova	University of Leeds, UK
Jouni Lampinen	University of Vaasa, Finland
Xiaodong Li	RMIT University, Australia
John McCall	Robert Gordon University, UK
Ernesto Mininno	University of Jyväskylä, Finland
Yew Soon Ong	Nanyang Technological University of Singapore, Singapore
Zhang Qingfu	University of Essex, UK
William Rand	University of Maryland, USA
Khaled Rasheed	University of Georgia, USA
Hendrik Richter	HTWK Leipzig University of Applied Sciences, Germany
Philipp Rohlfshagen	University of Birmingham, UK
Kay Chen Tan	National University of Singapore, Singapore
Ke Tang	University of Science and Technology of China, China
Yoel Tenne	Sydney University, Australia
Renato Tinos	Universidade de Sao Paulo, Brazil
Ville Tirronen	University of Jyväskylä, Finland
Shengxiang Yang	University of Leicester, UK
Gary Yen	Oklahoma State University, USA

EvoTRANSLOG Programme Committee

Christian Blum	Univ. Politecnica Catalunya, Spain
Peter A.N. Bosman	Centre for Mathematics and Computer Science, The Netherlands
Marco Caserta	University of Hamburg, Germany
Loukas Dimitriou	National Technical University of Athens, Greece
Karl Doerner	University of Vienna, Austria

Andreas Fink	Helmut Schmidt University Hamburg, Germany
Martin Josef Geiger	Helmut Schmidt University Hamburg, Germany
Stefan Irnich	RWTH Aachen University, Germany
Philippe Lacomme	University Blaise Pascal, Clermont-Ferrand, France
Mohamed Reghioui	University Abdelmalek Essaadi, Tetouan, Morocco
Franz Rothlauf	University of Mainz, Germany
Kay Chen Tan	National University of Singapore, Singapore
Theodore Tsekeris	Center of Planning and Economic Research, Greece
Stefan Voß	University of Hamburg, Germany
Oliver Wendt	University of Kaiserslautern, Germany

Sponsoring Institutions

- The University of Torino - School for Biotechnologies and Molecular Biotechnology Center, Torino, Italy
- The Human Genetics Foundation of Torino (HuGeF)
- The Museum of Human Anatomy ("Luigi Rolando"), Torino, Italy
- The Museum of Criminal Anthropology ("Cesare Lombroso"), Torino, Italy
- The Centre for Emergent Computing at Edinburgh Napier University, UK

Table of Contents – Part II

EvoTRANSLOG Contributions

Table of Contents – Part I

Investigation of Hyper-Heuristics for Designing Survivable Virtual Topologies in Optical WDM Networks

Fatma Corut Ergin, A. Şima Uyar, and Ayşegül Yayimli

Istanbul Technical University
fatma.ergin@marmara.edu.tr,
{etaner,gencata}@itu.edu.tr

Abstract. In optical WDM networks, a fiber failure may result in a serious amount of data loss, hence, designing survivable virtual topologies is a critical problem. We propose four different hyper-heuristic approaches to solve this problem, each of which is based on a different category of nature inspired heuristics: evolutionary algorithms, ant colony optimization, simulated annealing, and adaptive iterated constructive search are used as the heuristic selection methods in the hyper-heuristics. Experimental results show that, all proposed hyper-heuristic approaches are successful in designing survivable virtual topologies. Furthermore, the ant colony optimization based hyper-heuristic outperforms the others.

Keywords: Optical networks, WDM, survivable virtual topology design, hyper-heuristics.

1 Introduction

In today's world, the steady increase in user demands of high speed and high bandwidth networks causes researchers to seek out new methods and algorithms to meet these demands. The most effective medium to transmit data is the fiber. Optical networks [8] are designed for the best usage of the superior properties of the fiber, e.g. high speed, high bandwidth, physical strength, etc. Today, with the help of the wavelength division multiplexing (WDM) technology, hundreds of channels can be built on a single fiber. WDM is a technology in which the optical transmission is split into a number of non-overlapping wavelength bands, with each wavelength supporting a single communication channel operating at the desired rate. Since multiple WDM channels can coexist on a single fiber, the huge fiber bandwidth can be utilized.

A wavelength-routed WDM network provides end-to-end optical connections between two nodes in the network that are not necessarily connected directly by a fiber in the physical layer. These optical connections are called lightpaths. Two nodes become virtually neighbors when a lightpath is set up between them. More than one lightpath, each operating on different wavelengths, can be routed on the same fiber. All the lightpaths set up on the network form the virtual

© Springer-Verlag Berlin Heidelberg 2011

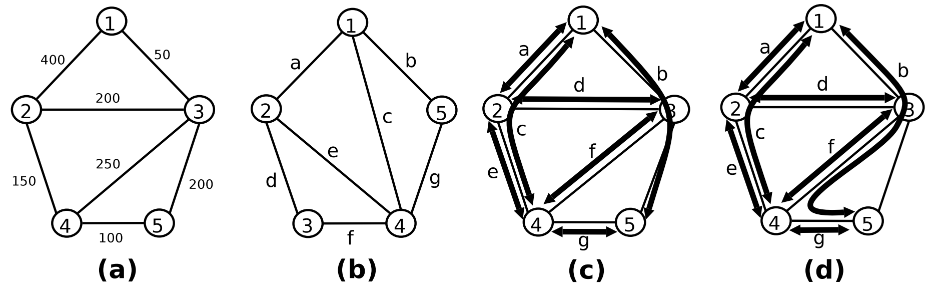


Fig. 1. a. Physical Topology, b. Virtual Topology, c. Survivable Mapping, d. Unsurvivable Mapping

topology (VT). Given the physical parameters of the network (physical topology, optical transceivers on the nodes, number of wavelengths that can be carried on the fibers, etc.) and the mean traffic intensities between the nodes, the problem of determining the lightpaths to be set up on the physical topology is known as the VT design problem.

In a WDM network, failure of a link (fiber) may result in the failure of several lightpaths routed through this link, which leads to several terabits of data loss. Survivable VT design aims to provide a continuous connectivity, using less resources. The continuous connectivity is ensured by designing the VT such that the VT remains connected in the event of a single link failure.

Assume that we have a physical network topology as in Figure 1.a and the virtual network topology to be routed on this physical topology is designed as in Figure 1.b. To obtain a survivable design of this VT, the mapping may be as in Figure 1.c. In this survivable mapping, a single 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 communicate and node 5 will be disconnected from the rest of the network.

Survivable VT design consists of four subproblems: determining a set of lightpaths (forming the VT), routing these lightpaths on the physical topology, so that any single fiber cut does not disconnect the VT, assigning wavelengths, and routing the packet traffic. Each of these subproblems can be solved separately. However, they are not independent problems and solving them one by one may degrade the quality of the final result considerably. Furthermore, the survivable VT design 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. In this study, we try to solve the survivable VT design problem as a whole using different hyper-heuristic (HH) [1] approaches. A HH is a method used to select between the low-level heuristics (LLH) at each step of an optimization process. This way, the best features of different heuristics can be combined.

In this study, we propose four HH approaches to design a survivable VT for a given physical topology, while minimizing resource usage. The proposed HHs use four different methods for heuristic selection: evolutionary algorithms (EA) [4], simulated annealing (SA) [6], ant colony optimization (ACO) [3], and adaptive iterated constructive search (AICS) [6]. From these methods, SA and EA are perturbative search methods, while AICS and ACO belong to the group of constructive search algorithms. Furthermore, SA and AICS are single point search methods, whereas, EA and ACO work on a population of solution candidates.

The rest of the paper is organized as follows: The survivable VT design problem is defined in Section 2. In Section 3, a detailed explanation of the approaches we propose to solve the problem is given. The experimental results are presented in Section 4. Section 5 concludes the paper.

2 Survivable Virtual Topology Design Problem

In optical networks, any damage to a physical link (fiber) on the network causes all the lightpaths routed through this link to be broken. Since huge amounts of data (e.g. 40 Gb/s) can be transmitted over each of these lightpaths, a fiber damage may result in a serious amount of data loss. Several different protection mechanisms have been proposed [7] for the fiber and/or other network equipment failures. Basically, there are two approaches for the fiber failure: 1) Survivability on the physical layer 2) Survivability on the virtual layer.

In the first approach, each connection passing through the fiber, i.e. the lightpath, is protected by assigning backup lightpaths that are disjointly routed from the connection's first lightpath. On the other hand, the second approach ensures the VT to be connected even in the failure of any single physical link. The first approach provides survivability by providing extra routes for each lightpath in the VT, however, with a cost of a high number of unemployed network resources. Thus, it offers an expensive solution for applications which may not need a high level of protection. The second approach, which has attracted attention especially in recent years, is a cost effective solution. Today, most applications are tolerant to latencies of several minutes of repair time needed by the packet layer (web search, file transfer, messaging, etc.), as long as the network connection is not terminated. This approach uses less network resources than the first one, thus, it enables service providers to offer a more economic service to their users.

Solving the survivable VT design problem as a whole is NP-complete [2]. Therefore, most of the previous studies on this problem consider only the survivable VT mapping subproblem [5]. There are only two studies in literature which try to solve all the subproblems together: a tabu search heuristic [9] and an ILP formulation [2]. The tabu search heuristic in [9] is constrained with small nodal degrees (up to 5) of VTs. Similarly, because of the problem complexity, the ILP method in [2] can solve very small problem instances of up to 4 node physical topologies, optimally.

2.1 Formal Problem Definition

The survivable VT design problem is defined as follows:
Given:

- Physical topology: Nodes and physical links that connect the nodes,
- Average traffic rates between each node pair,
- Maximum number of lightpaths that can be established on a node, i.e. the number of transceivers per node,
- Lightpath bandwidth capacity

Find:

- A collection of lightpaths to be established as a VT
- A survivable mapping (routing of the lightpaths over the physical topology, and wavelength assignment)
- A suitable routing of the packet traffic over the VT

The detailed ILP formulation for the survivable VT design problem can be found in [2]. Based on this formulation, the objective is to minimize the resource usage of the network, i.e. the total number of wavelength-links used in the physical topology. A wavelength-link is defined as a wavelength used on a physical link. For example, in Figure 1.c, 2 wavelength-links are used on the link between nodes 1 and 2, 1 wavelength-link is used on the link between nodes 1 and 3, ..., and a total of 9 wavelength-links are used in the physical topology.

3 Proposed Solution to the Survivable Virtual Topology Design Problem

To solve the survivable VT design problem, we use four HH approaches, each of which is based on a different type of nature inspired heuristic (NIH), used as the heuristic selection method. These NIHs are: evolutionary algorithms (EA), ant colony optimization (ACO), adaptive iterated constructive search (AICS), and simulated annealing (SA). Each method belongs to a different category of search approaches:

1. **EA:** population based, perturbative search
2. **ACO:** population based, constructive search
3. **SA:** single point perturbative search
4. **AICS:** single point, constructive search

Given the traffic matrix, the first step is to *determine a suitable VT*. For this subproblem, we selected three commonly used VT design heuristics as LLHs. At each step of the solution construction, a LLH is used to choose the next set of node pairs to establish a lightpath in between. The first LLH chooses the nodes which have the maximum single direction traffic demand between them. The second LLH considers the maximum bidirectional total traffic demand between node pairs. The third LLH chooses a node pair randomly. These three LLHs will be abbreviated as MAX_SNG, MAX_TOT, and RND, respectively for the rest of the paper. The lightpaths are established, such that, the in and out degrees of the nodes do not exceed the maximum number of transceivers on each node. In a preliminary study, we used two more LLHs to determine the nodes to add a lightpath. The first one chooses the node pairs with the minimum single direction traffic demand between them and the second one chooses those with the minimum bidirectional total traffic demand in between. The results showed that these LLHs do not improve the solution quality. Therefore, in this study we do not work with these LLHs.

The *VT routing and wavelength assignment (mapping) subproblem* is solved in our previous work [5] using ACO, which provides high quality solutions in a relatively small amount of time. Therefore, we use the same ACO approach to solve this subproblem in this study, too.

Traffic routing is applied in a straightforward way. The shortest path routing heuristic [8] is used for routing the packet traffic.

In HHs, the quality of the solutions is determined through a fitness function. In this study, the fitness of a solution is measured as the total number of wavelength-links used throughout the network, which is referred to as resource usage. The objective of the survivable VT design problem is to minimize this resource usage while considering the survivability and the capacity constraints. Resource usage is calculated by counting the number of physical links that are used by the lightpaths. An infeasible solution can either be penalized by adding a value to the fitness function, or can be discarded. In our HH algorithms, if a solution is found to be infeasible during the phases of creating a VT and routing the lightpaths, it is discarded. In the next stage, if the traffic cannot be routed over the physical topology, a penalty value proportional to the amount of traffic that cannot be routed, is added to the fitness of the solution.

A solution candidate is represented as an array of integers showing the order of LLHs to select lightpaths to be established. Since there are 3 different LLHs, the integers can have values between 1 and 3. When a lightpath between two selected nodes is established, the lightpath is assumed to be bidirectional. Therefore, a tranceiver at both ends is used. The length of the solution array is equal to the maximum number of lightpaths that can be established, i.e. $number\ of\ transceivers\ on\ each\ node * number\ of\ nodes/2$. For example, in a network with 6 nodes and 3 transceivers per node, each solution candidate is of length 6*3/2=9. If a solution candidate is represented with an array of [2 1 1 3 2 3 2 2 2], this means that, first a lightpath will be selected using the second LLH, then the next two using the first, continuing with the third, second, ... LLHs. While adding the lightpaths, the transceiver capacity constraint is handled. If the lightpath added according to the corresponding LLH results in using more than the existing number of transceivers in one or both ends, this lightpath is not added to the VT and the algorithm continues with the next LLH in the solution array. The lightpath capacity is 40 Gb/s and the traffic flows between the nodes is routed through the established lightpath between them.

Algorithm 1. General flow for the survivable VT design algorithm

Input: physical topology and traffic matrix
Output: a survivable VT

```
1: repeat
2:     use HHs to generate a VT
3:     use ACO to find a survivable mapping of lightpaths on the physical topology
4:     use shortest path heuristic to route traffic on the physical topology
5:     calculate fitness of the solution candidate
6:     apply NIH operators
7: until a predefined number of solutions are generated
```

The algorithm establishes lightpaths until either the end of the solution array is reached or until no traffic remains in the traffic matrix.

For each solution candidate produced by the HH, the corresponding VT is determined using the method explained above. Then, if the generated VT is not at least 2-connected (at least 2 link-disjoint paths for packet traffic exist between each node pair), new lightpaths are added subject to the transceiver capacity until the VT becomes 2-connected. For the nodes that have a degree lower than two, a new lightpath is added between this node and the node with the highest traffic demand in between. Next, the best mapping for the VT is found using ACO [5]. Then, the packet traffic is routed through the shortest paths starting from the node pair with the largest traffic demand. Finally, the fitness is calculated as the total amount of resource usage, i.e. the number of wavelength-links used throughout the network. The general flow of the algorithm is given in Algorithm 1.

3.1 Evolutionary Algorithms as a HH

We use a steady-state EA with duplicate elimination. After generating an initial set of random solution candidates, the EA operators, i.e. tournament selection, uniform crossover, and gene mutation, are applied and new solution candidates are generated. Gene mutation is defined as changing a LLH in the selected point of the string with another randomly determined LLH. Initial population of solution candidates (individuals) is generated randomly.

3.2 Ant Colony Optimization and Adaptive Iterated Constructive Search as HHs

ACO and AICS are very similar in the way they solve the problem. We can say that AICS is a form of ACO which uses only one ant. We use the elitist ant system (EAS) as the ACO variation, based on the results of our previous study [5], which show that this variation performs better than the others on the survivable VT mapping subproblem. However, there is only one ant in AICS, and the AS is applied as the ACO variation.

Initially, each ant iteratively adds a random LLH to its partial solution. The solution construction terminates when a solution array with a length equal to the maximum number of lightpaths is generated. No constraint is applied to the solution in the construction phase. Since there is no heuristic information, the solution construction only depends on the pheromone trail. The pheromone trails τ_{ij} we use in this paper refer to the desirability of using the j^{th} LLH to add the i^{th} lightpath. Pheromone trails are initialized using the initial random solutions of the ants. Then, they are modified each time that all ants have constructed a solution.

3.3 Simulated Annealing as a HH

We use a non-standard SA where the neighborhood operator is modified over time. The neighborhood operator is defined similar to the mutation operator in the EA-based HH, where with a given mutation probability, a randomly chosen LLH on the solution candidate is replaced by another LLH. The difference is that, we define a larger mutation probability in the beginning of the SA. The mutation probability is decreased by a predefined factor each time after 5 solution candidates are generated. This allows us to have a high exploration rate in the beginning of the search while focusing on exploitation towards the end. In our study, we used the formula given in [10] to calculate the initial temperature.

4 Experimental Results

We present the experimental results for a 24-node 43-link telco network [8], which is a fairly large-sized network for this problem. For the experiments, we use 20 different traffic matrices, randomly generated according to a frequently-used traffic generation method [8], where, 70% of the traffic is uniformly distributed over the range [0, 0.5 Gb/s] and 30% of the traffic is uniformly distributed over the range [0, 5 Gb/s]. The lightpath channel capacity is chosen as 40 Gb/s, which is typical in real-world networks.

First, we performed tests to see the performance of each LLH separately on the problem. For this test, only one LLH is used to generate the complete solution. Tests are run once for each LLH and traffic matrix pair.

Next, for the second group of experiments, we perform tests to find good parameter settings for the approaches used as heuristic selection methods in the HHs. We run the program 20 times for each tried parameter set.

As a result of parameter setting tests of *EA*, we selected 10 as the population size, and $1/l$, as the mutation probability, where l is the solution array length, and 0.8 as the crossover probability. The tests show that no significant improvement is obtained after a total of 100 individuals are created in the EA. Therefore, each run is terminated after 100 individuals are generated.

For the *AICS* parameters, the test results show that the value of q_0 does not affect the solution quality significantly. However, a slightly better solution quality is achieved with a q_0 value of 0.8. We selected 0.1 as the ρ value. To perform a fair comparison between methods, the termination condition is selected as creating 100 solution candidates. Since *ACO* is similar to *AICS* in which more than one ants are used, again, the q_0 value is selected as 0.8, the ρ value as 0.1, and the program is terminated after 100 solutions are generated. We selected 5 as the number of ants.

In *SA* parameter setting tests, the initial temperature selection is performed similar to [10]. As a result of the tests, we use an initial temperature of 195. The termination condition is again the same, i.e. 100 solution generations. The cooling rate is selected as 0.85, since this rate decreases the temperature to 5% of the initial temperature in 100 steps. The initial mutation probability is selected as $30/l$, where l is the solution array length, and is decreased with a factor of 0.85 in every 5 steps of solution generation. The large initial mutation probability is selected because of the large size of the solution array. If we start with a small mutation probability, the *SA* algorithm will search in a small neighborhood of the starting point, which may lead to getting stuck in local minima. The mutation rate is gradually decreased to avoid a random exploration in the search space. We decrease the mutation probability until it reaches a value of $1/l$.

After the parameter tuning of each approach, tests to explore their performance are conducted. In this set of experiments, we include a *Random* heuristic as a baseline for the performance comparisons. In *Random*, 100 random solution strings are generated, which are composed of randomly selected LLHs. For each traffic matrix, the solution candidate with the best fitness is selected as the solution. The tests are run 20 times for each traffic matrix. Each run uses a random initial seed. The results of the experiments are given in Table 1. Success rate of an approach is defined as the percentage of feasible solutions found. In the table, SR stands for success rate. The TM_i values in the first column of the table indicate that, the results in the corresponding row are the results obtained using

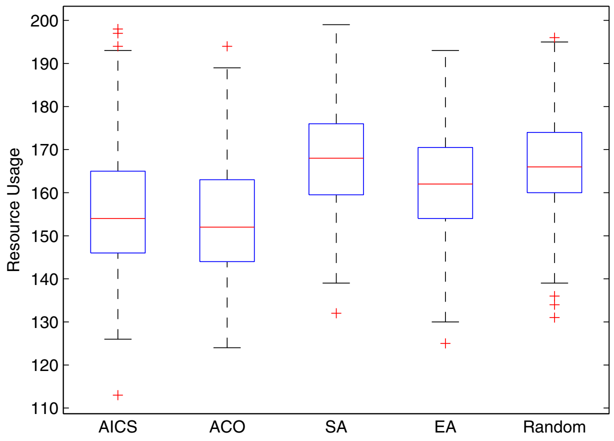


Fig. 2. The box-whisker plot of the results obtained using HHs and *Random* heuristic

traffic matrix i. In the next four columns of Table 1, the results for the four HHs designed in this study are given. The fifth column lists the results of the *Random* heuristic. The last three columns contain the resuls of LLHs applied separately. In the table, the values in the first five columns are the averages of resource usages obtained after 20 runs for each traffic matrix using the corresponding method. The last three columns are the results of using only the corresponding single LLH in the solution. The *NA* values in the table mean that a feasible solution was not found in any of the runs. The corresponding box-whisker plots are given in Figure 2.

From Table 1, we see that all the HHs perform better than the single LLHs. The success rates for the LLH_{RND}, *Random* and all the HHs are 100%, while, this rate is 70% for LLH_{MAX_SNG} and LLH_{MAX_TOT}. The best results obtained for each traffic matrix is marked in bold in the table. While in three of the traffic matrices, a single LLH produces the best result, it should be noted that the success rate for these LLHs is not 100%. The results show that, *ACO* finds the best result for 13 out of 20 traffic matrices, while the next method is *AICS* with 5 best results. To test the statistical significance of the results of the HHs and *Random*, we applied a two-way ANOVA and a Tukey HSD post hoc

Table 1. Resource usage results for different traffic matrices using different approaches

	EA	***ACO***	***AICS***	***SA***	***Random***	LLH_{MAX_SNG}	LLH_{MAX_TOT}	LLH_{RND}
TM_1	**169**	177	181	180	176	186	200	222
TM_2	158	152	149	163	165	**148**	160	229
TM_3	163	**148**	149	168	166	155	*NA*	219
TM_4	160	**146**	152	160	162	172	172	226
TM_5	154	**149**	151	156	157	170	159	220
TM_6	172	**156**	**156**	173	170	183	188	212
TM_7	175	**169**	172	185	180	*NA*	**169**	203
TM_8	167	**159**	162	173	173	167	169	222
TM_9	157	157	**151**	166	163	*NA*	*NA*	235
TM_{10}	172	161	**159**	176	174	194	198	221
TM_{11}	156	**137**	145	165	161	*NA*	155	222
TM_{12}	158	146	**145**	164	164	149	*NA*	218
TM_{13}	157	158	158	156	**155**	182	*NA*	236
TM_{14}	150	**135**	**135**	156	154	152	170	229
TM_{15}	168	**148**	155	176	173	*NA*	161	236
TM_{16}	163	**145**	147	165	165	*NA*	*NA*	219
TM_{17}	178	**175**	182	177	182	195	188	215
TM_{18}	**152**	162	156	160	163	*NA*	*NA*	213
TM_{19}	154	**139**	140	156	157	178	180	223
TM_{20}	170	**163**	166	175	177	**163**	177	221
Average	163	154	156	168	167	171	175	222
SR	1.0	1.0	1.0	1.0	1.0	0.7	0.7	1.0

test at a confidence level of 0.95. The results show that *ACO* and *AICS* produce results which are statistically significantly better than *SA*, *EA*, and *Random*, while a statistically significant difference cannot be observed between *ACO* and *AICS*. Therefore, we can say that, constructive search techniques are more successful for this problem. Furthermore, based only on averages, we conclude that a population based scheme is preferable to a single point one.

5 Conclusion

The HH approaches proposed in this study for survivable VT design is able to solve the problem for large-sized networks having intensive traffic demands with a 100% success rate. The results show that, using HHs can combine the best features of the LLHs and give better results. Furthermore, the HH approach based on ACO outperforms the other HH approaches based on EA, AICS and SA. This suggests that, a population based, constructive search approach as a HH is more suitable for this problem. As future work, the ILP solution to the problem with small-size networks will be investigated and results of the HHS will be compared to the optimum values.

References

1. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: Handbook of Meta-heuristics. Springer, Heidelberg (2010)
2. Cavdar, C., Yayimli, A.G., Mukherjee, B.: Multilayer resillient design for layer-1 VPNs. In: Optical Fiber Communication Conference and Exposition - OFC 2008, California, USA (February 2008)
3. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
4. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
5. Ergin, F.C., Kaldirim, E., Yayimli, A., Uyar, A.S.: Ensuring resilience in optical WDM networks with nature inspired heuristics. IEEE/OSA Journal of Optical Communications and Networking 2(8), 642–652 (2010)
6. Hoos, H.H., Stutzle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann, San Francisco (2005)
7. Kurant, M., Nguyen, H.X., Thiran, P.: Survey on dependable IP over fiber networks. Research Results of the DICS Program, 55–81 (2006)
8. Mukherjee, B.: Optical WDM Networks. Springer, Heidelberg (2006)
9. Nucci, A., Sanso, B., Crainic, T.G., Leonardi, E., Marsan, M.A.: On the design of fault-tolerant logical topologies in wavelength-routed packet networks. IEEE Journal on Selected Areas in Communications 2(9), 1884–1895 (2004)
10. Topcuoglu, H., Ermis, M., Corut, F., Yilmaz, G.: Solving the uncapacitated hub location problem using genetic algorithms. Computers & Operations Research 32(4), 967–984 (2005)

On Improving the Capacity of Solving Large-scale Wireless Network Design Problems by Genetic Algorithms*

Fabio D'Andreagiovanni

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)
Takustrasse 7, 14195 Berlin, Germany
d.andreagiovanni@zib.de

Abstract. Over the last decade, wireless networks have experienced an impressive growth and now play a main role in many telecommunications systems. As a consequence, scarce radio resources, such as frequencies, became congested and the need for effective and efficient assignment methods arose. In this work, we present a *Genetic Algorithm* for solving large instances of the *Power, Frequency and Modulation Assignment Problem*, arising in the design of wireless networks. To our best knowledge, this is the first Genetic Algorithm that is proposed for such problem. Compared to previous works, our approach allows a wider exploration of the set of power solutions, while eliminating sources of numerical problems. The performance of the algorithm is assessed by tests over a set of large realistic instances of a *Fixed WiMAX Network*.

Keywords: Wireless Network Design, Large-scale Optimization, Genetic Algorithms.

1 Introduction

During the last years, wireless communications have experienced an explosive growth thus rapidly leading to a dramatic congestion of radio resources. In such complex scenario, the trial-and-error approach commonly adopted by practitioners to design networks has clearly shown its limitations. Telecommunications companies and authorities are thus searching for more effective and efficient design approaches, also looking on Optimization (as shown by the recent call for tenders for developing a Digital Video Broadcasting simulator by the Italian Communications Regulatory Authority [2]). Many models and solution methods have been proposed for solving the problem of designing a wireless network. However, solving to optimality the overall problem is still a big challenge in the case of large instances. In this paper we present a Genetic Algorithm for solving the *Power, Frequency and Modulation Assignment Problem*, a relevant problem that arises in the design of wireless networks and captures specific features of Next Generation Networks.

* This work was supported by the *German Federal Ministry of Education and Research* (BMBF), project *ROBUKOM*, grant 03MS616E.

© Springer-Verlag Berlin Heidelberg 2011

2 The Wireless Network Design Problem

For modeling purposes, a wireless network can be described as a set of transmitters B that provide for a telecommunication service to a set of receivers T. A receiver $t \in T$ is said to be *covered* or *served* if it receives the service within a minimum level of quality. Transmitters and receivers are characterized by a location and a number of radio-electrical parameters (e.g., power emission and frequency). The *Wireless Network Design Problem* (WND) consists in establishing the location and suitable values for the parameters of the transmitters with the goal of maximizing the number of covered receivers or a revenue associated with coverage.

In this work we consider a generalization of the so-called *Power and Frequency Assignment Problem (PFAP)*, a version of the WND that is known to be NP-hard [12]. In addition to power emission and frequency, we also consider the transmission scheme (*burst profile*) as parameter to be established, modeling a feature of Next Generation Networks, such as WiMAX [3,6]. We indicate this generalization of the PFAP by the name *Power, Frequency and Modulation Assignment Problem (PFMAP).*

In the PFMAP two decisions must be taken: (1) establishing the power emission of each transmitter on each available frequency and (2) assigning served receivers to activated transmitters specifying the frequency and the burst profile used to transmit. To model these decisions, as first step we introduce the set F of available frequencies and the set H of available burst profiles. Each frequency $f \in F$ has a constant bandwidth D and each burst profile $h \in H$ is associated with a *spectral efficiency* s_h, which is the bandwidth required to satisfy one unit of traffic demand generated by a receiver.

Then we can introduce two typologies of decision variables, namely:

– a continuous power variable $p_b^f \in [0, P_{\max}]$ $\forall\, b \in B, f \in F$ representing the power emission of a transmitter b on frequency f;

– a binary service assignment variable $x_{tb}^{fh} \in \{0,1\}$ $\forall\, t \in T, b \in B, f \in F, h \in H$ defined in the following way:

$$x_{tb}^{fh} = \begin{cases} 1 & \text{if receiver } t \in T \text{ is served by transmitter } b \in B \\ & \text{on frequency } f \in F \text{ through burst profile } h \in H \\ 0 & \text{otherwise} \end{cases}$$

Given a frequency $f \in F$, every receiver $t \in T$ picks out signals from every transmitter $b \in B$ and the power $P_b^f(t)$ that t receives from b on f is proportional to the emitted power p_b^f by a factor $a_{tb} \in [0,1]$, i.e. $P_b^f(t) = a_{tb} \cdot p_b^f$. The factor a_{tb} is called *fading coefficient* and summarizes the reduction in power that a signal experiences while propagating from b to t [14].

Among the signals received from transmitters in B, a receiver t can select a *reference signal* (or *server*), which is the one carrying the service. All the other signals are interfering. We remark that, since each transmitter in B is associated with a unique signal, in what follows we will also refer to B as the set of signals received by t.

A receiver t is regarded as served by the network, specifically by server $\beta \in B$ on frequency $f \in F$ through burst profile $h \in H$, if the ratio of the serving power to the sum of the interfering powers (*signal-to-interference ratio* or *SIR*) is above a threshold δ_h (*SIR threshold*) whose value depends on the used burst profile h [14]:

$$\frac{a_{t\beta} \cdot p_\beta^f}{N + \sum_{b \in B \setminus \{\beta\}} a_{tb} \cdot p_b^f} \ \geq \ \delta_h \ . \tag{1}$$

Note that in the denominator we highlight the presence of the system noise $N > 0$. By simple algebra operations, inequality (1) can be transformed into the following linear inequality, commonly called *SIR inequality*:

$$a_{t\beta} \cdot p_\beta^f - \delta_h \sum_{b \in B \setminus \{\beta\}} a_{tb} \cdot p_b^f \ \ \geq \ \ \delta_h \cdot N \ . \tag{2}$$

As we do not know a priori which transmitter $b \in B$ will be the server of a receiver $t \in T$ and which frequency $f \in F$ and burst profile $h \in H$ will be used, given a receiver $t \in T$ we have one SIR inequality (2) for each potential server $\beta \in B$ and potentially usable frequency f and burst profile h. To ensure that t is covered, at least one of such inequalities must be satisfied. This requirement can be equivalently expressed through the following disjunctive constraint:

$$\bigvee_{\forall (\beta,f,h): \beta \in B, f \in F, h \in H} \left(a_{t\beta} \cdot p_\beta^f - \delta_h \sum_{b \in B \setminus \{\beta\}} a_{tb} \cdot p_b^f \ \ \geq \ \ \delta_h \cdot N \right) \ . \tag{3}$$

Adopting a standard approach used in Mixed-Integer Programming (see [13]), the above disjunction can be represented by a family of linear constraints in the p variables by introducing a large positive constant M, the so-called *big-M coefficient*. Specifically, given a receiver $t \in T$ we use the assignment variable $x_{t\beta}^{fh}$ to introduce the following constraint for each potential 3-ple (β, f, h):

$$a_{t\beta} \cdot p_\beta^f - \delta_h \sum_{b \in B(t) \setminus \{\beta\}} a_{tb} \cdot p_b^f \ \ + \ \ M \cdot (1 - x_{t\beta}^{fh}) \ \ \geq \ \ \delta_h \cdot N \ . \tag{4}$$

It is easy to check that if $x_{t\beta}^{fh} = 1$ then (4) reduces to the simple SIR constraint (2). If instead $x_{t\beta}^{fh} = 0$ and M is sufficiently large[1], then (4) is satisfied by any feasible power vector p and becomes redundant.

By using constraints (4) and by introducing a parameter r_t to denote revenue associated with receiver $t \in T$ (e.g., population, number of customers) , we can define the following natural formulation (BM-PFMAP) for the PFMAP [6,7]:

$$\max \quad \sum_{t \in T} \sum_{b \in B} \sum_{f \in F} \sum_{h \in H} r_t \cdot x_{tb}^{fh} \tag{BM-PFMAP}$$

[1] For example, we can set $M = \delta_h \cdot N + \delta_h \sum_{b \in B \setminus \{\beta\}} a_{tb} \cdot P_{max}$.

$$\text{s.t.} \quad a_{t\beta} \cdot p_{\beta}^{f} - \delta_h \sum_{b \in B \setminus \{\beta\}} a_{tb} \cdot p_b^f + M \cdot (1 - x_{t\beta}^{fh}) \geq \delta_h \cdot N \qquad t \in T, b \in B, f \in F, h \in H \quad (5)$$

$$\sum_{b \in B} \sum_{f \in F} \sum_{h \in H} x_{tb}^{fh} \leq 1 \qquad t \in T \quad (6)$$

$$\sum_{t \in T} \sum_{h \in H} d_t \cdot \frac{1}{s_h} \cdot x_{t\beta}^{fh} \leq D \qquad b \in B, f \in F \quad (7)$$

$$p_b^f \in [0, P_{max}] \qquad b \in B, f \in F \quad (8)$$

$$x_{tb}^{fh} \in \{0,1\} \qquad t \in T, b \in B, f \in F, h \in H \quad (9)$$

The objective function is to maximize the total revenue obtained by serving receivers and constraint (6) ensures that each receiver is served at most once. Each receiver generates a traffic demand d_t and each frequency has a bandwidth equal to D. Constraint (7) ensures that the sum of traffic demands (re-sized by the spectral efficiency s_h of the used burst profile) generated by the receivers served by a transmitter does not exceed the bandwidth of the frequency. Finally, (8) and (9) define the decision variables of the problem.

Drawbacks of the natural formulation. The natural formulation (BM-PFMAP) expands a basic model that is widely used for the WND in different application contexts, such as DVB, (e.g., [12]), UMTS (e.g., [1,11]) and WiMAX ([6,7]). In principle, such basic model and (BM-PFMAP) can be solved by commercial solvers such as IBM ILOG CPLEX [10]. However, it is well-known (see [7]) that: (i) the fading coefficients may vary in a wide range leading to (very) ill-conditioned coefficient matrices that make the solution process numerically unstable; (ii) the big-M coefficients generate poor quality bounds that dramatically reduce the effectiveness of standard solution approach [13]; (iii) the resulting coverage plans are often unreliable and may contain errors (e.g., [7,11]). In practice, the basic model and (BM-PFMAP) can be solved to optimality only when used for small-sized instances. In the case of large real-life instances, even finding feasible solutions can represent a difficult task, also for state-of-the-art commercial solvers like CPLEX. Though these drawbacks are well-known, it is interesting to note that just a relatively small part of the wide literature devoted to WND has tried to overcome them. We refer the reader to [6] for a review of works that have tried to tackle these drawbacks.

2.1 Contribution of This Work and Review of Related Literature

In this paper, we develop our original contribution by starting from a recent work, [7], that proposes a family of strong valid inequalities for tackling the drawbacks of (BM-PFMAP) that we have pointed out. The idea at the basis of [7] is to quit modeling emission power as a continuous variable p_b and to use instead a set of discrete power levels $\mathcal{P} = \{P_1, \ldots, P_{|\mathcal{P}|}\}$, with $P_1 = 0$ (*switched-off value*), $P_{|\mathcal{P}|} = P_{max}$ and $P_i > P_{i-1}$, for $i = 2, \ldots, |\mathcal{P}|$. This basic operation allows the authors to define a family of *lifted GUB cover inequalities* that are used in a solution algorithm that drastically enhances the quality of solutions found.

The solution algorithm proposed in [7] is motivated by a trade-off that arises from discretization: larger sets of discrete levels lead in principle to better solutions, but on the other hand the corresponding 0-1 Linear Program gets larger and harder to solve. The computational experience shows that very good solutions can be found by considering small sets with well-spaced power values, but that no improvement is obtained within the time limit when a number of levels higher than six is considered.

In the present work, we investigate the possibility of using a *Genetic Algorithm (GA)* [8] as a fast heuristic to widen the exploration of the discrete power solution space: the aim is to exploit the entire set of discrete power levels and thus to evaluate power configurations with levels not included in the best solutions found in [7]. In particular, our aim is to improve the capacity of solving large realistic instances by finding higher value solutions. We thus design a GA that takes into account the specific features of the PFMAP and we test its performance on the same set of realistic WiMAX instances used in [7].

Heuristics have been extensively used to tackle large instances of different versions of the WND problem. Two relevant cases are provided by [1], where a two-stage Tabu Search algorithm is proposed to solve the base station location and power assignment problem in UMTS networks, and by [12], where a GRASP algorithm is proposed to solve the PFAP arising in the planning of the Italian National DVB network. The use of GA to solve versions of the WND is not a novelty as well and many works can be found in literature. However, to our best knowledge, no GA has been yet developed to solve the PFMAP and the algorithm that we propose is the first for solving this level of generalization of the WND. Until now, GAs were indeed developed to solve just single aspects of the PFMAP: (i) the transmitter location problem (e.g., [4]); (ii) the service assigment problem (e.g., [9]); (iii) the frequency assignment problem (e.g., [5]); (iv) the power assignment problem (e.g., [15]). Moreover, we remark that our algorithm is the first to be designed with the specific aim of improving the capacity of solving instances, while tackling the numerical problems pointed out in Section 2. We now proceed to present our original contributions for the WND.

3 A Genetic Algorithm for the PFMAP

A Genetic Algorithm (GA) is a heuristic method for solving optimization problems that resembles the evolution process of a population of individuals (for a comprehensive introduction to the topic we refer the reader to [8]). At any iteration, a GA maintains a population whose individuals represent feasible solutions to the problem. The solution is encoded in a *chromosome* associated with each individual. The *genetic strength* of an individual is evaluated by a fitness function that establishes the quality of the corresponding solution to the problem.

A GA starts by defining an initial population, that iteration after iteration changes by crossover, mutation and death of individuals, according to a natural selection Darwinistic mechanism.

We develop a GA for the PFMAP that presents the following general structure:

1. Creation of the initial population
2. UNTIL the arrest condition is not satisfied DO
   (a) Selection of individuals who generate the offspring
   (b) Generation of the offspring by crossover
   (c) Mutation of part of the population
   (d) Death of part of the population

We now characterize the elements and the phases presented above for the algorithm (GA-PFMAP) that we propose to solve the PFMAP.

3.1 Characterization of the Population

Individual representation. As we have explained in Section 2.1, our aim is to conduct a wider exploration of the power solution space, trying to obtain solutions with higher value. To this end, we establish that the chromosome of an individual corresponds to a power vector p of size $|B| \cdot |F|$. Specifically, the chromosome presents one *locus* for each transmitter $b \in B$ and frequency $f \in F$ and each locus stores the power p_b^f emitted by b on f, namely $p = (p_1^1, \ldots, p_1^{|F|}, p_2^1, \ldots, p_2^{|F|}, \ldots, p_{|B|}^{|F|})$. Such power belongs to the set of discrete power levels $\mathcal{P}$, i.e. $p_b^f \in \mathcal{P} = \{P_1, \ldots, P_{|\mathcal{P}|}\}$.

We remark that establishing the power emission $p_b^f \;\; \forall\, b \in B,\;\, f \in F$ does not completely characterize a solution of the PFMAP. We indeed have to fix the value of the assignment variables x_{tb}^{fh} and thus we need to set some assignment rule. First, note that given a power vector $p = (p_1^1, p_1^2, \ldots, p_{|B|}^{|F|})$ and a receiver $t \in T$ we can compute the power $P_b^f(t)$ that t receives from b on f, $\;\forall b \in B, f \in F$. Through $P_b^f(t)$, if we fix the server $\beta \in B$ of t, we can check if there exists a SIR inequality (2) that is satisfied for some frequency $f \in F$ and burst profile $h \in H$. We establish the following assignment rule: as server of t we choose the transmitter b that ensures the highest received power $P_b^f(t)$ on some f. This in fact ensures the highest serving power. Once that the server β is chosen, we can identify the SIR inequalities (2) that are satisfied by p when t is served by β for some $f \in F$ and $h \in H$. If the SIR inequality is satisfied for a multiplicity of frequencies and/or burst profiles, we first choose as serving frequency $\hat{f}$ the one that ensures the highest value for the left-hand-side of (2) and then we choose as burst profile $\hat{h}$ the one that ensures the highest spectral efficiency (see Section 2). Thus for t served by β we have $x_{t\beta}^{\hat{f}\hat{h}} = 1$ and $x_{t\beta}^{fh} = 0 \;\forall f \in F \setminus \{\hat{f}\}, h \in H \setminus \{\hat{h}\}$.

Note that this last rule may assign a receiver t to a transmitter β that violates the capacity constraint (7) of β on the chosen frequency $\hat{f}$. If this is the case, we choose the second best couple of frequency and burst profile according to the rule. If this not possible, the third best and so on. In the end, if there is no capacity left for any valid couple (f, h), t is not considered covered by β.

Fitness function. As the aim of the WND is to maximize coverage, we adopt a fitness function that evaluates the coverage ensured by an individual. Specifically, the fitness $COV(p)$ of an individual is equal to the number of receivers that are covered when the power vector is p and service assignment is done according to the previously introduced rules.

Initial population. Our aim is to consider all the feasible discrete power levels from the beginning. Therefore, the initial population is represented by the power vectors that are obtained by activating a single transmitter $b \in B$ on a single frequency $f \in F$ at each of the discrete power levels $P_l \in \mathcal{P}$. For every $b \in B, f \in F$, the corresponding individuals included in the initial population are thus: $(0, 0, \ldots, p_b^f = P_2, \ldots, 0, 0) \quad \cdots \quad (0, 0, \ldots, p_b^f = P_{|\mathcal{P}|}, \ldots, 0, 0)$. Note that we exclude the individual corresponding to all transmitters turned off, i.e. $p_b^f = P_1 = 0 \; \forall b \in B, f \in F$. We thus have $|B| \cdot |F| \cdot |L - 1|$ initial individuals. We denote the set of individuals representing the population at a generic iteration of the algorithm by P.

3.2 Evolution of the Population

Selection. In order to select the individuals who give birth to the new generation, we adopt a *tournament selection* approach: given the set P of individuals constituting the current population and a value $0 < \alpha < 1$, we first define a number $k \in \mathbb{Z}^+$ of groups including $\lfloor \alpha \cdot |P| \rfloor$ individuals who are randomly extracted from P. Then we extract $m < \lfloor \alpha \cdot |P| \rfloor$ individuals with the best fitness from every group. These are the individuals who generate offspring by crossover.

Crossover, mutation and death. The individuals selected for crossover are randomly paired up to constitute $\lfloor k \cdot m/2 \rfloor$ couples. Each couple generates two offspring by mixing its chromosome. Given a couple of parents with power vectors $p1, p2$, the crossover operation consists in mixing power levels that are in the same position of $p1$ and $p2$ to generate two offspring with (possibly) higher fitness power vectors $p3, p4$.

Before presenting the crossover procedure, we need to define a measure that evaluates the impact of crossover on coverage. To this end, let $\Delta\mathrm{COV}(p, p_b^f = P_l) \in \mathbb{Z}$ denote the variation in the number of covered receivers caused by changing the power value p_b^f in position (b, f) of vector p to the value P_l, while maintaining all the other power values unchanged. We can then propose the following crossover procedure, that concentrates the effort of creating a higher fitness individual on $p3$. At the beginning of the crossover, $p3$ and $p4$ have all elements equal to 0. Then, by following this ordering of indices $(b, f) : b \in, f \in F$: $(1,1)$ $(1,2)$ $\ldots$ $(1, |F|)$ $(2,1)$ $\ldots$ $(2, |F|)$ $\ldots$ $(|B|, 1)$ $\ldots$ $(|B|, |F|)$, each null value inherits the power value in the same position of one of the two parents.

We now present the crossover rule for a generic position (β, ϕ). For indices $(b, f) : b < \beta, f < \phi$, the crossover was already executed and thus the offspring vectors $p3, p4$ present power levels inherited by the parents $p1, p2$. Power levels of $p3, p4$ in positions $(b, f) : b \geq \beta, f \geq \phi$ are instead still equal to zero. The rule to establish the power value inherited by $p3, p4$ in (β, ϕ) is the following:

$$p3_{\beta}^{\phi} = \begin{cases} p1_{\beta}^{\phi} & \text{if } \Delta\text{COV}(p3, p3_{\beta}^{\phi} = p1_{\beta}^{\phi}) \geq \Delta\text{COV}(p3, p3_{\beta}^{\phi} = p2_{\beta}^{\phi}) \\ p2_{\beta}^{\phi} & \text{otherwise} \end{cases}$$

$$p4_{\beta}^{\phi} = \begin{cases} p1_{\beta}^{\phi} & \text{if } \Delta\text{COV}(p3, p3_{\beta}^{\phi} = p1_{\beta}^{\phi}) < \Delta\text{COV}(p3, p3_{\beta}^{\phi} = p2_{\beta}^{\phi}) \\ p2_{\beta}^{\phi} & \text{otherwise} \end{cases}$$

This ensures that, at any step of the crossover procedure, offspring $p3$ inherits the power level of the parent that allows the most favourable variation ΔCOV in coverage.

In addition to crossover, we also allow to alter the power vector of single individuals by *mutation*. This introduces new genetic information in the population and helps to widen the solution space exploration and to avoid entrapment in local optima. At any iteration, a number of individuals $\lfloor \gamma \cdot |P| \rfloor$ with $0 < \gamma < 1$ is randomly chosen. Then, still by random selection, $|F|$ power levels corresponding with different frequencies are reduced to the immediately lower power level allowed in $\mathcal{P}$. This mutation rule is set with the aim of defining new power vectors that have lower powers but ensure the same coverage. The reduction in power is generally desirable as a signal that is useful for a receiver may be interfering for a different receiver.

Finally, after crossover and mutation, the weakest individuals *die* and are removed from P. Specifically, we choose to select and remove the $2 \cdot \lfloor k \cdot m/2 \rfloor$ individuals with the worst fitness function. The size of P is thus maintained constant over all the iterations.

4 Computational Experience

We test the performance of our GA on a set of 15 realistic instances, developed with the Technical Strategy & Innovations Unit of British Telecom Italia (BT Italia SpA). All the instances refers to a *Fixed WiMAX Network* [3], deployable in an urban area corresponding to a residential neighborhood of Rome (Italy). The instances consider various scenarios with up to $|T| = 529$ receivers, $|B| = 36$ transmitters, $|F| = 3$ frequencies, $|H| = 4$ burst profiles (see Table 1). This leads to large formulations (BM-PFMAP) that are very hard to solve. For a detailed description of the instances, we refer the reader to [7].

For each instance, we run the proposed algorithm (GA-PFMAP) 50 times with a time limit of 1 hour by using a machine with a 1.80 GHz Intel Core 2 Duo processor and 2 GB of RAM. Each tournament selection involves $k = 20$ groups that include a fraction $\alpha = 0.05$ of the population P. The best $m = 8$ individuals of each group are selected for crossover and, after the generation of the new individuals, mutation affects a fraction $\gamma = 0.1$ of the population.

In Table 1, we compare the value of the best solution obtained through the three approaches that we consider, namely the direct solution of (BM-PFMAP) by ILOG Cplex 10.1, the solution of the Power-Indexed formulation by the algorithm WPLAN [7] and the solution of (BM-PFMAP) by the proposed algorithm (GA-PFMAP). Results for (BM-PFMAP) and WPLAN are derived from [7].

Table 1. Comparisons between (BM) and WPLAN formulations

ID	|T|	|B|	|F|	|H|	|T*| (BM-PFMAP)	|T*| WPLAN [7]	|T*| (GA-PFMAP)
S1	100	12	1	1	63 (78)	74	70
S2	169	12	1	1	99 (100)	107	103
S3	196	12	1	1	108	113	113
S4	225	12	1	1	93	111	115
S5	289	12	1	1	77	86	88
S6	361	12	1	1	154	170	175
S7	400	18	1	1	259 (266)	341	319
R1	400	18	3	4	370	400	400
R2	441	18	3	4	302 (303)	441	441
R3	484	27	3	4	99 (99)	427	434
R4	529	27	3	4	283 (286)	529	462
Q1	400	36	1	4	0	67	72
Q2	441	36	1	4	191	211	222
Q3	484	36	1	4	226	463	466
Q4	529	36	1	4	145 (147)	491	491

The presence of two values in some lines of the column of (BM-PFMAP) indicates that the coverage plans returned by Cplex contain errors and some receivers are actually not covered. We remark that (GA-PFMAP) provides solutions that always ensure a higher coverage than (BM-PFMAP) and without coverage errors. Making a comparison with WPLAN, we instead note that (GA-PFMAP), though in some cases finds solutions with lower coverage, for most of the cases is able to find solutions that ensure an equal or higher number of covered receivers than WPLAN. This is particularly evident for instances that seems to be very hard to solve through (BM-PFMAP). The algorithm is thus effective and is worth of further investigations.

5 Conclusion and Future Work

We presented a Genetic Algorithm (GA) to tackle large realistic instances of a relevant problem arising in wireless network design. We showed that a GA helps to improve the value of solutions found through a wider exploration of the power space. A future research path could be represented by the integration of a refined GA into an exact solution method. It is indeed common experience that the combination of fast heuristics with Mixed-Integer Linear Programming leads to a great reduction in the running times w.r.t pure exact optimization methods. A sensitivity analysis of the GA parameters and a study on the impact of different starting conditions and selection strategies would also constitute important subjects of investigations.

Acknowledgments. The author thanks Carlo Mannino and Antonella Nardin for fruitful discussions. Thanks go also to the three anonymous referees who provided valuable comments.

References

1. Amaldi, E., Belotti, P., Capone, A., Malucelli, F.: Optimizing base station location and configuration in UMTS networks. Ann. Oper. Res. 146(1), 135–152 (2006)
2. Autoritá per la Garanzia nelle Comunicazioni, Delibera N.269/09/CONS, `http://www.agcom.it/default.aspx?DocID=3359`
3. Andrews, J.G., Ghosh, A., Muhamed, R.: Fundamentals of WiMAX. Prentice Hall, Upper Saddle River (2007)
4. Choi, Y.S., Kim, K.S., Kim, N.: The Displacement of Base Station in Mobile Communication with Genetic Approach. EURASIP J. Wireless Comm. and Net. (2008)
5. Colombo, G.: A Genetic Algorithm for Frequency Assigment with Problem Decomposition. Int. J. Mob. Net. Des. and Innov. 1(2), 102–112 (2006)
6. D'Andreagiovanni, F.: Pure 0-1 Programming Approaches to Wireless Network Design. Ph.D. Thesis, Sapienza Università di Roma, Rome, Italy, Winner of the 2010 INFORMS Doctoral Dissertation Award for Operations Research in Telecommunications (2010)
7. D'Andreagiovanni, F., Mannino, C., Sassano, A.: GUB Covers and Power-Indexed Formulations for Wireless Network Design. Technical Report vol. 2 n. 14, Department of Computer and System Sciences, Sapienza Università di Roma, Rome, Italy (2010)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading (1988)
9. Hu, T., Chen, Y.P., Banzhaf, W.: WiMAX Network Planning Using Adaptive-Population-Size Genetic Algorithm. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 31–40. Springer, Heidelberg (2010)
10. IBM ILOG CPLEX, `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer`
11. Kalvenes, J., Kennington, J., Olinick, E.: Base Station Location and Service Assignments in W-CDMA Networks. INFORMS J. Comp. 18(3), 366–376 (2006)
12. Mannino, C., Rossi, F., Smriglio, S.: The Network Packing Problem in Terrestrial Broadcasting. Oper. Res. 54(6), 611–626 (2006)
13. Nehmauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley & Sons, Hoboken (1988)
14. Rappaport, T.S.: Wireless Communications: Principles and Practice, 2nd edn. Prentice Hall, Upper Saddle River (2001)
15. Song, W.J., et al.: Evolutionary Computation and Power Control for Radio Resource Management in CDMA Cellular Radio Networks. In: PIMRC 2002, vol. 3, pp. 1417–1421 (2002)

Dynamic Routing Exponent Strategies for Ant-Based Protocols*

Rui Fang[1], Zequn Huang[2], Louis F. Rossi[1], and Chien-Chung Shen[2]

[1] Mathematical Sciences
{ruifang,rossi}@math.udel.edu
[2] Computer and Information Sciences
University of Delaware, Newark, DE 19716, USA
{zehuang,cshen}@cis.udel.edu

Abstract. In ant-based routing protocols, the routing exponent controls how ants hop from node to node to discover routes based on pheromone values. It has been shown that stable multi-route solutions for small routing exponent values are dynamically connected to stable single-route solutions for large routing exponent values. These stable single-route solutions correspond to paths that have the smallest hop count. In this paper, we leverage this idea to improve the performance of ant-based routing protocols by dynamically adjusting the routing exponent. The results are validated via simulation.

1 Introduction

Swarm intelligence is a term that refers to the action of a locally coordinated group of individuals that can achieve a complex objective or behavior. Often the local coordination algorithms are inspired by ecological systems including social insects, self-organizing colonies of single-celled organisms or movements of larger animals such as flocks of birds. Each individual possesses incomplete information about the problem to be solved, and coordination is achieved typically through interaction with a subset of individuals in the swarm. Through these interactions, complex, near-optimal behavior can emerge [1]. One successful application of swarm intelligence is the use of ant-based protocols to route data through networks.

Ant-based routing protocols use control packets, called "ants," to explore networks, discover routes and reinforce the best routes. Throughout this paper, we will use the terms "ant" and "control packet" interchangeably. True ants in the biological world mark foraging trails with chemical pheromone that can be detected by other ants. The pheromone evaporates over time, so that inefficient

* This work is supported by U.S. Army Research Laboratory contract #W911QX-09-C-0076. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

© Springer-Verlag Berlin Heidelberg 2011

routes fade from disuse. More efficient popular routes are reinforced as ants deposit more pheromone along them. Similarly, in ant-based protocols, virtual pheromone is stored on the nodes as the ants traverse the network. In short, the key to the routing protocol is a spatially distributed, mathematical model of pheromone deposition and evaporation. Research has shown that ant-based routing protocols provide an effective solution to the routing problem of both wired networks [2] and mobile ad hoc networks [3–5]. In this paper, we will use a mathematical framework for studying routing protocol dynamics to improve their performance.

A modeling framework introduced in [6] to describe the evolution of pheromones in ant-based routing protocols using dynamical systems theory correctly predicts stationary states of realistic network protocols. In this study, it was shown that some of the principles gained from rigorous analysis of small networks, transfer to larger networks that are much more difficult to be mathematically analyzed. In particular routing exponents that are much smaller than unity, lead to multi-route solutions and exponents that are much larger than unity lead to single-route solutions. However, not all single route solutions are optimal in the sense that they require more than the minimum number of hops to travel from a sender node to a receiver node. Also, it was shown that if one treats the routing exponent β as a parameter, stable multi-route solutions were dynamically connected to the optimal single route solution on small networks. In this paper, we will leverage this idea to develop a strategy, analogous to simulated annealing, to improve the performance of ant-routing protocols on large networks by dynamically adjusting the routing exponent.

This paper is organized as follows. In Section 2, we describe a modeling framework, which is the mathematical foundation of dynamic routing exponent strategy. In Section 3, we introduce this strategy by two specific examples, a 5-node network and a 50-node network using Matlab and QualNet simulations, and then validate its effectiveness by performing experimental comparisons. Section 4 concludes the paper.

2 Preliminaries of Ant-Based Routing

In this paper, a network is viewed as a directed graph. Each link is weighted by pheromone values, which determine how ants will travel in the network along multi-hop routes. Using pheromone tables on each node, ant-based routing protocols deploy ants to discover possible routes between pairs of nodes, and optimize routing tables to enhance shorter, desirable routes via pheromone deposition and discard longer, less efficient routes via evaporation of pheromone. In our mathematical modeling framework, the behaviors of ant-based routing are characterized by three general rules: route discovery, route reinforcement (deposition) and route decay (evaporation).

Route discovery is accomplished by the random motion of ants through the network as they hop from node to node. An ant at node i will move to node j with probability p_{ij}

$$p_{ij} = \frac{(\tau_{ij})^{\beta}}{\sum_{h \in N_i} (\tau_{ih})^{\beta}} \tag{1}$$

where τ_{ij} represents the pheromone values on the link from node i to node j, N_i is the set of all connected neighbors of nodes i and β is the routing exponent. The protocol uses two different types of ants. Ants traveling from source s, seeking route to destination d. are called "forward ants." If the network consists of m nodes, we define $\mathbf{y}^{(n)}$ be the m-dimensional vector probability density of ants over the network at the n^{th} time step. The forward ants traverse the network following the Markov process according to a transition matrix $P^{(n)}(\beta) = [p_{ji}]$ at the n^{th} time step,

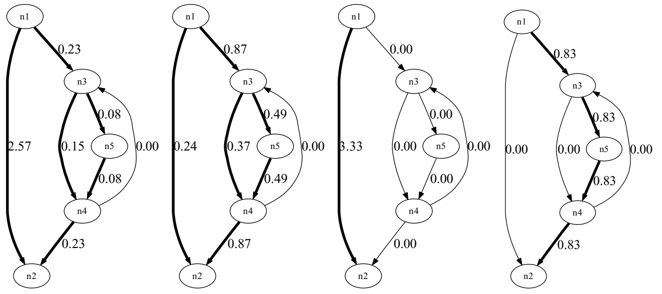
$$\mathbf{y}^{(n+1)} = P^{(n)}(\beta)\mathbf{y}^{(n)} \tag{2}$$

because both probability density and pheromone values on each link are evolving only depend on present state by every discrete synchronous step. Here the k^{th} component of the density vector $\mathbf{y}^{(n)}$ is the probability of finding an ant on the k^{th} node of the network. Once a forward ant reaches the destination, it becomes a "backward ant," and will trace back to the source from the destination, depositing pheromone along the route it takes. The routing protocol defines how the matrix $P = [p_{ji}]$ evolves from one iteration to the next through pheromone deposition and evaporation. We denote the matrix P at discrete time step n as $P^{(n)}$. From the previous analysis, we know that the routing exponent β controls whether single-path routes are selected or multi-path routes are selected. For a complete description and analysis of the protocol, see [6]. In this section, we will review the essential features and properties of the protocol.

A full-fledged routing protocol is very difficult to analyze because it has many parameters and features to respond to different contingencies. Instead, we will study and implement a very simple routing protocol and explore it using an analytic framework. Since ant-based routing is a dynamic process, we identify two critical time increments. The increment h_1 is the amount of time between evaporation events on each node. The increment h_2 is the amount of time between deployment of ants from a source node s. We assume that $h_1 \leq h_2$. The routing protocol can be described as follows:

1. N ants are released from the source node. The source node resets its clock to $t = 0$.
2. Each ant moves through the network following (1) and maintaining a node-visited stack until it arrives at the destination node.
3. An ant reaching the destination node will retrace its steps back to the source. If the ant's route from source to destination is cycle-free (i.e. no node is visited twice), the ant deposits pheromone along the links it traverses. Otherwise, no pheromone is deposited. Typically, the amount of pheromone deposited on a link is inversely proportional to the hop count of the route traveled between source s and destination d.
4. When a backward ant arrives at the source, it is destroyed.
5. When the source node clock reaches $t = h_2$, return to step 1.

Independent of all the ant activity, pheromone will decay along all links.



(a) S1 (b) S1p (c) S5 (d) S7

Fig. 1. Stationary states calculated using the stochastic model. Solutions S1, S5 and S7 were calculated with $\beta = 0.5$ and $\Lambda = 0.3$. Solution S1p was calculated with $\beta = 2$ and $\Lambda = 0.3$.

In [6], an analytic model for the behavior of this ant-based routing protocol is derived:

$$\tau_{ij}^{(n+1)} = \underbrace{(1 - h_1\kappa_1)^{(h_2/h_1)}\tau_{ij}^{(n)}}_{\text{evaporation}} + \underbrace{h_2\kappa_2 \sum_{k=1}^{\infty} \frac{1}{k^p}\tilde{p}_{ij}^{sd}(k)}_{\text{deposition}}, \tag{3}$$

where κ_1 is an evaporation rate, κ_2 is a deposition rate and $\tilde{p}_{ij}^{sd}(k)$ is the probability of an ant following a k-hop route from source node s to destination node d passing through link ij without any cycles. The link undergoes h_2/h_1 evaporation events between step 1 and step 5, and it is understood that many transitions of (2) occur for every one transition of (3). Also, the summation is the expected inverse hop count, $\left\langle \frac{1}{H_{sd}} \right\rangle$ for a single ant. This is a natural way to reinforce shorter routes more than longer routes because the amount of pheromone deposited along the route is inversely proportional to the path cost.

If we think of the ant-based protocol as a nonlinear dynamical system, we can understand network performance by examining stationary solutions and their linear stability. A stationary state occurs when (2) and (3) are independent of the time step n and satisfy the system,

$$\Lambda\tau_{ij} = \sum_{k=1}^{\infty} \frac{1}{k}\tilde{p}_{ij}^{sd}(k) \tag{4}$$

where τ_{ij} is an equilibrium pheromone distribution and $\Lambda = \frac{\kappa_1}{\kappa_2}$ is called pheromone deposition number. Note that $\tilde{p}_{ij}^{sd}(k)$ depends upon τ_{ij}. Since this is a nonlinear system, solutions are not necessarily unique. In fact, a single β, Λ pair may have many stationary solutions. The eigenvalues and eigenvectors of the Jacobian of this system reveals whether or not a given stationary solution is stable. Earlier work presented a phase diagram for a representative 5-node network [6].

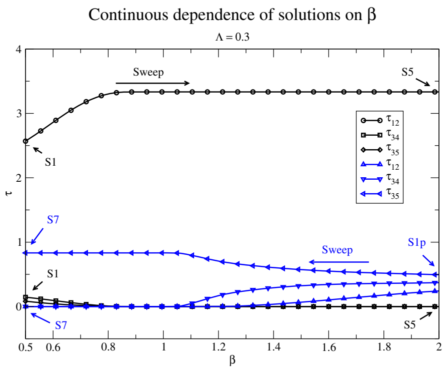


Fig. 2. Continuous dependence of solutions on β for the simple 5-node network configuration used in Figure 1

The previous work shows that some of the principles and features gained from rigorous study of small networks are consistent with larger and more complicated networks that are much more difficult to analyze. In particular, for the simple 5-node network and a larger 50-node network, small routing exponents $\beta \ll 1$ lead to stable, multi-route solutions whereas large exponents $\beta \gg 1$ lead to stable single-route solutions. Examples are shown in Figure 1 where solution S1 is stable but solutions S1p, S5 and S7 are unstable in the parameter regimes used to generate the solutions. However, solutions with the same qualitative structure as S5 and S7 are stable when $\beta = 2$.

Moreover, stable multi-route solutions are dynamically connected to the optimal single route solution on the 5-node network. As shown in Figure 2, if we follow the structure of certain stationary solutions, we see that the stable multiple-route solution S1 is dynamically connected to the optimal single-route solution S5. On the other hand, the unstable multiple route solution S1p is connected to the suboptimal, unstable single-route solution S7. One possible explanation is that shorter routes are reinforced more readily when the system is more deterministic which is the case when β is large.

These results and observations on large networks suggest that we can improve ant-based routing by dynamically adjusting the routing exponent. Large values of β offer the advantage of stability and determinism, but there is no guarantee that the single-route selected will be optimal. The earlier study suggests that the system will naturally evolve into the optimal single-route state if we start with a stable multi-route solution with β small and then steadily increase β to a large value.

3 Dynamic Routing Exponents

In this section, we first validate the efficiency of this technique of dynamically adjusting the routing exponent on a 5-node network. Then, we leverage this idea to show that on large networks, it is also possible to improve the performance of ant-routing protocols. We implement our algorithm in Matlab, without a physical communication model so there are no packet drops, and in QualNet with realistic protocol and communication models. Table 1 summarizes the parameter settings used by both Matlab and QualNet simulations. In the QualNet simulation, each topology is modeled as a point-to-point mesh network with link bandwidth of 100 Mbps and link propagation delay of 1 millisecond. The ant-based routing protocol operates in the network layer, and encapsulates ants in the IP packets.

Table 1. Table of parameters used in network simulations

Total time of Simulation	199.99
N	200
β	$0.5 \rightarrow 2$
κ_1	0.3
κ_2	1
Λ	0.3
h_1	1
h_2	1

3.1 5-Node Network

To demonstrate the concept of using dynamic routing exponents, we apply this idea on the 5-node network where the dynamics are well understood. (We note that a phase portrait was calculated in [6]). We initiate simulation with random pheromone values over the network. Rather than using a fixed value, β will slowly vary from 0.5 to 2.0 as follows.

$$\beta = \begin{cases} 0.5, & t < 50 \quad \text{(allow multiroute solution to stabilize)} \\ 0.5 + \frac{t-50}{20}, & 50 \leq t \leq 80 \quad \text{(move network toward single route sol'n)} \\ 2.0, & t > 80 \quad \text{(proceed with optimal solution)} \end{cases} \qquad (5)$$

This function allows the routing protocol to relax into a multiroute solution before slowly moving the system toward an optimal or near-optimal single-route solution. Thus, we expect the network to move toward S1 when $0 < t \leq 50$ and then move toward S5. This is precisely what we observe in Figure 3, which demonstrates dynamic pheromone distribution on four critical routes of the simple 5-node networks shown in Figure 1.

3.2 50-Node Network

With the same configurations summarized in Table 1, the Matlab simulation for a 50-node network has successfully validated the anticipated result that routing exponent $\beta = 0.5$ leads to the formation of multiple route solution, while $\beta = 2$, on the other hand, corresponds to the existence of the single route solutions. We consider optimal solutions to be solutions with the minimum possible hop-count. Single route stable solutions found with large routing exponents are observed to be close to optimal, but not typically optimal. In Figure 4, we see two typical

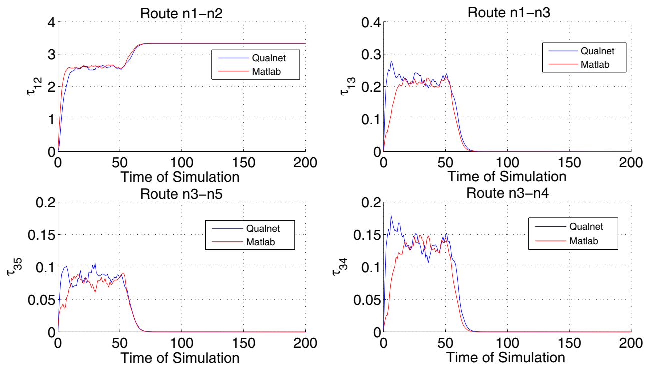


Fig. 3. Pheromone values as a function of time using dynamic routing exponents for both Matlab and QualNet simulations

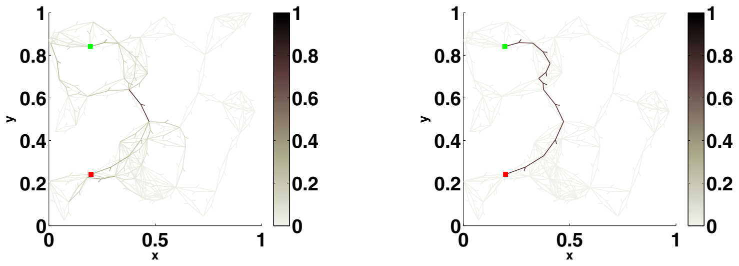


Fig. 4. Two stationary solutions with $\beta = 0.5$ on the left, $\beta = 2$ on the right, both with $\Lambda = 0.3$. Node s is shown by red spot and node is shown by green spot. Pheromone values are normalized by the maximum pheromone value over the entire network.

stable solutions. Since the shortest path possible connecting source and destination in this network is 6-hops, the single route is nearly optimal (7 hops) but not optimal.

Now we will explore our dynamic routing exponent strategy (5) by translating the design principles for the small network to 50-node network settings. Again, the simulations begin with random pheromone values. Among the total time of simulation of ant-based routing protocol, we capture several key instants, shown by Figure 5, that illustrate the reorganization of pheromone values over the network driven by the dynamic routing exponent β. Similar to the 5-node network case, the system initially settles into a multi-route solution as shown in Figure 5(a). As β increases, the network evolves toward single route solutions with intermediate stages shown in Figures 5(b,c). Beyond this point, the system is in a stable, optimal, 6-hop solution as shown in Figure 5(d). In addition, Figure 6 depicts one instance of the evolution of average hop count over time as β changes linearly from 0.5 to 2 between simulation time instances 50 and 80.

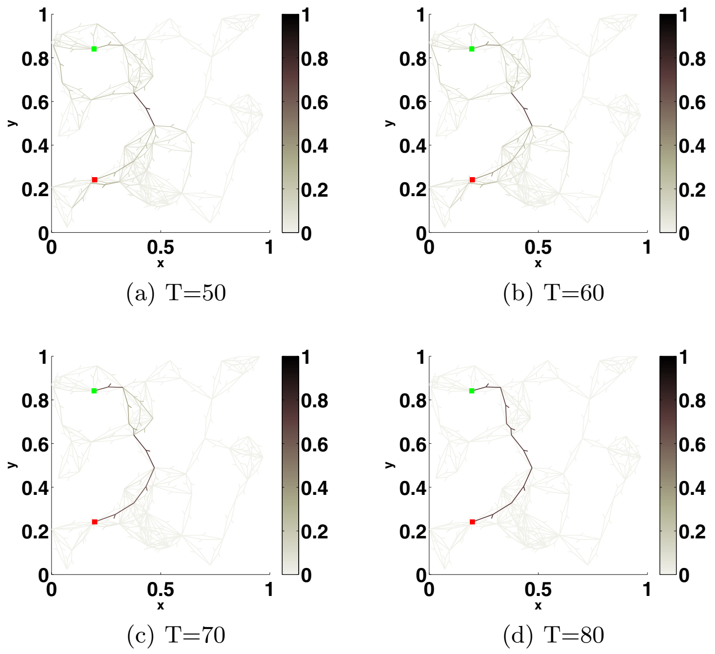


Fig. 5. Distribution of normalized pheromone values at some important moments of simulation. At time 50, a multi route state is achieved and beta starts to vary. The solution settles down at time 80, when β stops to vary.

3.3 Experimental Comparison and Impact of Λ

We implemented an empirical analysis to quantify how the performance of ant-based routing protocols can be improved by dynamically adjusting routing exponent from a value less than unity to that larger than unity. We performed the same experiment on the 50-node network 100 times using different random initial conditions and compared the performance using $\beta = 2$ (standard practice for protocols like AntHocNet) to using the dynamic β strategy. All other network parameters are the same. All the simulations are executed with random initial pheromone distribution and pheromone deposition number $\Lambda = 0.3$ (recall $\Lambda = \frac{\kappa_1}{\kappa_2}$). Our simple comparison, shown in Figure 7, demonstrates a considerable benefit when using the dynamic β strategy. The difference between the average length of a single route solution for the $\beta = 2$ strategy compared to the variable β strategy at the end of simulation is almost 2 hops (mean value: $8.01 \rightarrow 6.11$) for the Matlab simulations and 4 hops (mean value: $10.2 \rightarrow 6.2$) for the QualNet simulations.

Our experimental results indicate that the dynamic β strategy is still somewhat dependent on initial pheromone levels though not nearly so much as the constant β strategy. If the system is too far from a stable multi-route solution that includes an optimal path, the dynamic β strategy will not be able to

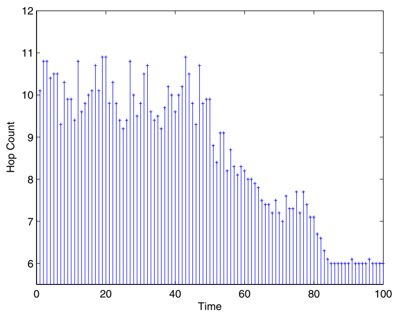


Fig. 6. Evolution of average hop count over time

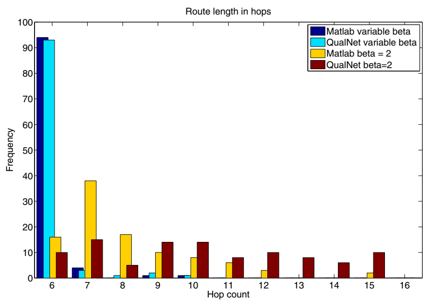


Fig. 7. Experimental study of dynamic routing exponent strategy on improving performance of routing protocols

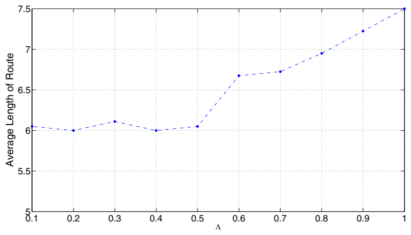


Fig. 8. Average length of optimal single route respect to Λ

smoothly move the network into an optimal configuration. This explains why the mean remains above 6 in the dynamic β trials in Figure 7.

Finally, we test the performance of dynamic routing exponent technique under different values of Λ, an equally crucial parameter as β in our model. Since κ_1 corresponds to the rate of evaporation while κ_2 corresponds to the rate of deposition, Λ controls the ratio of evaporation to deposition. We anticipate larger values of Λ to reduce the impact of deposition and so reduce the benefit of the overall technique as we try to drive the system toward an optimal solution. To illustrate this, we performed the experimental study on the 50-node network for

$\Lambda = 0.1, 0.2, \ldots, 1.0$. Figure 8 shows that when $\Lambda < 0.5$, the results are acceptable as the average length of the optimized single route is roughly 6. However, when $\Lambda > 0.5$, the average hop count of the optimized single routes increases linearly and reach 7.5 when $\Lambda = 1$, which is consistent with our expectation.

4 Conclusions

We have introduced and explored a new strategy that dynamically adjusts the routing exponent in ant-based protocols. The new strategy was motivated by an analytic framework that provided a complete description of the nonlinear dynamics of a small network. From this description, we observe that stable multi-route solutions for small β are dynamically connected to stable single-route solutions for large β. These stable single-route solutions correspond to paths that have the smallest hop count. We give two examples using simulations in both Matlab and QualNet, a simple, 5-node network and a larger 50-node network. For the 5-node network, the results are exactly compatible with the previous rigorous study. For the 50-node network, we leverage principles from the simpler 5-node network and successfully validate them via Matlab and QualNet simulation. In particular, we find the dynamic β protocol performs significantly better than the static $\beta = 2$ protocol in a large series of tests using random initial pheromone values. Finally, we explore the impact of Λ, the key parameter that determines the relative importance of evaporation and deposition. As expected, we find that the effectiveness of the dynamic β strategy will be impaired when Λ is large. However, for moderate Λ, our new dynamic β routing protocol finds shorter routes than traditional ant-based routing methods.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence – From Natural to Artificial Systems. Oxford University Press, New York (1999)
2. Di Caro, G.A., Dorigo, M.: AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
3. Di Caro, G.A., Ducatelle, F., Gambardella, L.: AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks. European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking 16(5), 443–455 (2005)
4. Ducatelle, F., Di Caro, G.A., Gambardella, L.: Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. Swarm Intelligence 4(3) (2010) (in press)
5. Rajagopalan, S., Shen, C.C.: ANSI: A Swarm Intelligence-based Unicast Routing Protocol for Hybrid Ad hoc Networks. Journal of System Architecture, Special issue on Nature Inspired Applied Systems 52(8-9), 485–504 (2006)
6. Torres, C.E., Rossi, L.F., Keffer, J., Li, K., Shen, C.C.: Modeling, analysis and simulation of ant-based routing protocols. Swarm Intelligence 4(3), 221–244 (2010)

Ant-Based Multipath Routing for Wireless Mesh Networks

Laurent Paquereau and Bjarne E. Helvik

Centre for Quantifiable Quality of Service in Communication Systems*,
Norwegian University of Science and Technology, Trondheim, Norway
{laurent.paquereau,bjarne}@q2s.ntnu.no

Abstract. Wireless Mesh Networks (WMNs) are envisioned as a flexible alternative for providing Internet access. In this context, one of the key challenges is to improve the capacity. One approach is to spread the load along multiple paths. Results achieved in wired networks using ant-based systems for this purpose make them attractive candidates. However, applying similar techniques directly to WMNs may be counter-productive due to the characteristics of multi-hop wireless communications, in particular interferences. In this paper, a novel hybrid approach, based on recording the Internet gateway used by ants and marking pheromone trails accordingly, is presented. Results are promising and indicate that adaptive and efficient load distribution can be achieved.

1 Introduction

Wireless Mesh Networks (WMNs) are multi-hop wireless backhaul networks providing Internet connectivity through a limited number of gateways. WMNs are envisioned as broadband access networks and differ in many respects from other types of multi-hop wireless networks, e.g. Mobile Ad-hoc NETworks (MANETs). Nodes are static, grid-powered, and may be equipped with multiple interfaces. The traffic is expected to be concentrated along paths to/from gateways, and the traffic volume to be high. In this context, the provision of high performance, dependability and security is a major concern. Exploiting to this end the inherent redundancy of such networks is a challenging research issue [1,16].

Multipath routing is one approach to take advantage of the path diversity in a network. It has been applied to various types of networks and is addressed in a large body of literature. See [15,19] and the references therein for examples of multipath routing protocols for wireless networks. It consists, as opposed to single path routing, in finding, maintaining and using multiple paths between source and destination pairs to improve performance, dependability and/or security. In particular, and this is the focus of this paper, increased capacity may be obtained by spreading the load along multiple paths. However, this requires some policies to select paths and efficiently distribute the traffic amongst them.

* "Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence" appointed by The Research Council of Norway, funded by the Research Council, NTNU, UNINETT and Telenor. http://www.q2s.ntnu.no

© Springer-Verlag Berlin Heidelberg 2011

Ant-based routing protocols are intrinsically multipath and provide, with pheromone trail values, a basis to select paths and split the traffic. Ant-based routing refers to the application of the Ant Colony Optimization (ACO) meta-heuristic [9] to path management in telecommunication networks. In this context, ants are control packets that are used to repeatedly sample paths between source and destination pairs. At each node, the pheromone trails reflect the knowledge acquired by the colony. The pheromone trail value associated with a given neighbour indicates the relative goodness of this neighbour to reach a specific destination. Good solutions emerge as the result of the collective pheromone trail laying and following behaviour of ants. Ant-based systems are distributed, adaptive and robust and, hence, constitute an attractive alternative to traditional path management systems. See for instance [11] for a survey of existing ant-based routing protocols.

In AntNet[8], for instance, load distribution is achieved by applying, at each node and for each data packet, a random proportional forwarding decision rule based on the local pheromone distribution. In this way, the load is effectively distributed on multiple paths proportionally to their estimated quality, and a performance increase is observed compared to deterministic unipath routing. However, this method has been designed for wired networks and applying it directly to multi-hop wireless networks may be counter-productive due to the nature of wireless communications, i.e. limited bandwidth, shared medium, intra- and inter-flow interferences, etc. In particular, [10] suggests that deterministic best cost routing performs better in MANETs. The presence of multiple gateways makes the situation different in WMNs and opens up for new approaches.

In this paper, a novel hybrid policy for adaptive stochastic multipath load distribution based on pheromone trails in WMNs is introduced. It relies on letting ants mark pheromone trails according to the gateways they sample. These markings are then used to select neighbours, and the load is distributed amongst the selected neighbours proportionally to their relative pheromone trail value. Most of the previous works on ant-based routing for multi-hop wireless networks consider generic MANETs and compare the performance achieved to that of traditional MANET routing protocols. This work specifically targets WMNs and compares the achieved load distribution with the optimal distribution. Related work, outside the field of ant-based routing, includes gateway-driven probabilistic load-balancing schemes [14] and Wardrop routing in wireless networks [18].

The rest of this paper is organized as follows. Section 2 reviews existing approaches to data forwarding based on pheromone trails. Their limitations in the context of WMNs are then characterized in Sect. 3, and the novel hybrid policy is presented and evaluated. Finally, concluding remarks are given in Sect. 4.

2 Data Forwarding Based on Pheromone Trails

The ability of ants, as a colony, to find good solutions to complex problems relies on the indirect communication between ants mediated by pheromones. In nature, pheromones are volatile chemical substances laid by ants while walking that modifies the environment perceived by other ants. In the context of ant-based routing, ants are control packets and pheromones are values stored at each node in the network. Ants are used to find and maintain paths between source and destination pairs. Each ant performs a random search directed by the pheromone trails and deposits pheromones on the path it samples, establishing new trails or reinforcing existing ones. Similarly to what happens in nature, good solutions emerge as the result of the iterative indirect interactions between ants.

Pheromone trails reflect the knowledge acquired by the colony. The focus of this paper is on how to use this information to select neighbours and distribute the data traffic. Existing forwarding policies based on pheromones trails can be classified into two categories: (i) *Random Proportional forwarding (RP$_\kappa$)*, which is a stochastic multipath approach, and (ii) *Highest Pheromone trail value forwarding (HP)*, which is a deterministic unipath approach. A last option would be to apply a deterministic multipath forwarding policy, e.g. OMS [4]. However, to the best of our knowledge, such an approach has never been applied in the context of ant-based routing. In the following, $\tau_{t_v,vi}^{(s,d)}$ denotes the pheromone trail value at node v towards node i after t_v updates for the source and destination pair (s,d), and $\mathcal{N}_v^{(s,d)}$ the subset of neighbours i of node v for which $\tau_{t_v,vi}^{(s,d)} \geqslant \varphi$. $\varphi \geqslant 0$ denotes a cut-off level used to ignore trails with low pheromone concentrations. The superscripts (s,d) are hereafter omitted to improve readability.

2.1 Random Proportional Forwarding (RP$_\kappa$)

The RP$_\kappa$ forwarding policy follows closely the “ACO philosophy”. The forwarding rule used for data packets is similar to the one used for ants. Formally, the probability of forwarding a packet to node i at node v is given by:

$$p_{t_v,vi} = \frac{\tau_{t_v,vi}^{\kappa}}{\sum_{j \in \mathcal{N}_v} \tau_{t_v,vj}^{\kappa}}, \ \forall i \in \mathcal{N}_v \ , \tag{1}$$

where $\kappa \geqslant 0$ is a configuration parameter controlling the bias towards good solutions. Figure 1 illustrates the effect of κ. The higher κ is, the more the best neighbours are used. At the two ends of the scale, RP$_0$ corresponds to purely random (uniform) forwarding and RP$_\infty$ to stochastic equal-cost multipath forwarding[1]. For $\kappa < \infty$, no specific neighbours are selected; a data packet may be forwarded to any neighbour $i \in \mathcal{N}_v$ such that $\tau_{t,vi} > 0$.

RP$_\kappa$ is used by most of the ant-based routing protocols, and in particular by AntNet [8] and AntHocNet [12]. In AntNet, κ is chosen close to 1 (from 1.2 [8] to 1.4 in [7]). Ants are assumed to converge to the optimal solution, and the data traffic is distributed accordingly. The rationale for choosing $\kappa > 1$ is to avoid forwarding packets along low-quality paths (AntNet uses $\varphi = 0$). For AntHocNet, the value reported for κ varies from originally 2 [6] to reach 20 in [12]. This increase is supported by the results of a sensitivity analysis presented in [10], which suggest that better performance is achieved in MANETs as $\kappa \to \infty$.

[1] In [10], $\kappa \to \infty$ is used to refer to deterministic highest pheromone forwarding; this is true only in the case there are no two equally high pheromone trail values.

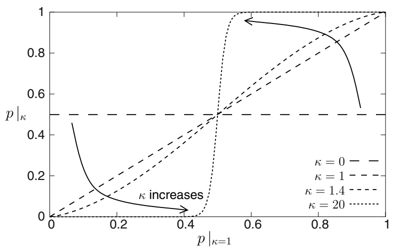


Fig. 1. Transformation of the forwarding probabilities in the case where $|\mathcal{N}_v| = 2$

2.2 Highest Pheromone Trail Value Forwarding (HP)

HP simply consists in choosing at each node v the neighbour i with the highest pheromone trail value, that is

$$i = \arg\max_{j \in \mathcal{N}_v} \tau_{t_v,vj} \quad . \tag{2}$$

Such a forwarding policy is used by PERA [2] for instance.

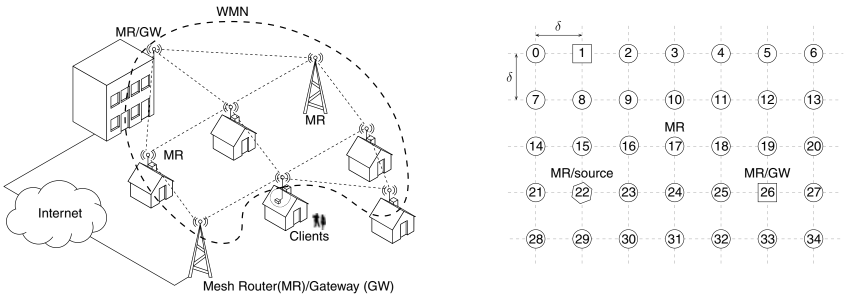
HP is a deterministic unipath forwarding policy. Note, however, that the output of (2) may vary after a pheromone update. Ant-based routing protocols applying HP are neither unipath nor static, but multipath adaptive routing protocols. Ants discover and maintain multiple paths, thus allowing the system to adapt to possible changes in terms of quality or topology. Simply, the same forwarding choice is made locally between two pheromone updates.

3 Exploiting Multiple Paths in Wireless Mesh Networks

3.1 Existing Forwarding Policies

In this section, the forwarding policies presented in Sect. 2 are evaluated by simulation in the context of WMNs. See Fig. 2(a) for an illustration. The objective is to get detailed insights and compare the different schemes in terms of achieved goodput and load distribution.

A homogeneous 802.11-based (no RTS/CTS), single channel, single rate, WMN is considered. Nodes are equipped with omni-directional antennas and organized in a regular grid as shown in Fig. 2(b). A two-ray ground radio propagation model is used, and the distance δ between nodes is chosen so that a given node is able to reach directly either 4 or 8 of the surrounding nodes. These reachability patterns are denoted $+$ and $\ast$, respectively. All the data traffic is assumed to be directed to or from the gateways. For this study, only a single Mesh Router (MR) (node 22) sends traffic towards a destination outside the WMN. The data traffic is generated by a Constant Bit Rate (CBR) source sending packets of 512 bytes over UDP. All the results reported below are averages of 20 replications and error bars indicate 95% confidence intervals.



(a) Architecture (b) Grid topology

Fig. 2. Infrastructure WMN

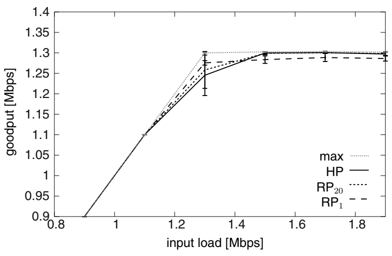
The ant-based system used for this evaluation is OCEAN[2], an opportunistic version of the Cross-Entropy Ant System (CEAS) [17]. See for instance [13] for a recent introduction to CEAS. OCEAN excludes low-quality neighbours from $\mathcal{N}_v$ by applying an adaptive pruning rule identical to the one used when forwarding ants ($\varphi > 0$); see [17] for details. This mechanism serves the same purpose as choosing κ slightly greater than 1 in AntNet. Therefore, $\mathrm{RP}_{1.4}$ is not considered in the comparison.

The cost metric used is the expected transmission count (ETX) [5]. The ETX of a link is an estimate of the number of transmissions needed to successfully transmit a unicast packet. It is computed as the inverse of the delivery ratio of dedicated locally broadcasted probes packets[3]. The ETX of path is the sum of the ETX of each link. Assuming perfect links, at low load, ETX is equivalent to hop-count. As the load gets closer to the capacity, ETX increases due to packet losses caused by interferences. ETX is one of simplest quality-aware cost metric proposed for WMNs. See for instance [3] for a survey of other proposed metrics.

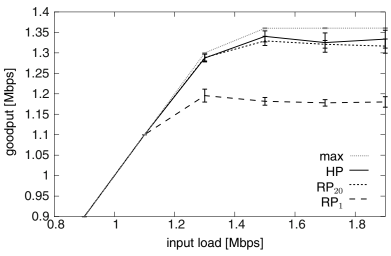
Single Gateway. Figure 3 shows the goodput achieved applying the different forwarding policies for different input loads when a single gateway (node 1) is available. Of interest here is the goodput achieved when the offered load exceeds the capacity. The maximum achievable goodput is included as a reference. Under saturated conditions, the highest goodput is obtained using only the best path, here $\langle 22, 15, 8, 1\rangle$ in both cases. In the $+$-case, Fig. 3(a), the goodput is similar for all forwarding strategies and close to the upper-bound. On the other hand, in the $\ast$-case, Fig. 3(b), the goodput achieved using RP_1 is significantly lower than the maximum and than the goodput obtained applying the other policies. The reason is the existence of a set of braided paths of equal length. Applying RP_1,

[2] The configuration parameters used for OCEAN are: $\beta = 0.95$, $\rho = 0.01$ and $\alpha = 0.1$. Ants are generated at the rate of 1 per second and 10% of the generated ants are explorer ants. See [13] and [17] for further details.

[3] In the experiments presented in this paper, nodes send one probe packet per second and delivery ratios are computed as simple moving averages over the last 20 seconds.



(a) +-reachability



(b) ∗-reachability

Fig. 3. Goodput (source: 22; gateway: 1)

the load is distributed on these paths, which increases the level of interferences and results in a decreased performance. Applying HP, a single path is used, thus limiting the amount of interferences and resulting in a goodput close to the maximum. Finally, using κ as high as 20 amounts here in practice in applying HP. These observations are in-line with the results for MANETs presented in [10].

Multiple Gateways. When multiple gateways are available, and especially when there exist disjoint, or partly disjoint, paths between the source and the gateways, the conclusions are different. Correctly distributing the load allows the system to achieve a higher goodput. Hence, the different policies are now evaluated not only with respect to the achieved goodput, but also to the achieved balance. Figures 4 and 5 show the results obtained when two gateways (nodes 1 and 26) are available. When the input load exceeds the capacity, in both cases, the highest goodput is obtained by sending traffic along $\langle 22, 15, 8, 1\rangle$ and $\langle 22, 23, 24, 25, 26\rangle$. The surface obtained by forwarding data packets stochastically on either of these paths is included as a reference. The best policy is the one closest to the ridgeline for all input loads. The 3D-view is shown in the +-case to ease the understanding. Results are otherwise presented using contour maps, Fig. 4(a) and 5(a), and section plots, Fig. 4(b) and 5(b), for representative input loads, namely 0.9, 1.3 and 2.5 Mbps. These loads correspond to the cases where either of the gateways, only gateway 1, and none of the gateways can be used alone to transmit all the input load, respectively. Results obtained applying the RS-policy are also included. This policy will be introduced in Sect. 3.2.

Applying HP, only the best path, and thus a single gateway, is used between two pheromone trails updates. Under saturated conditions, the system oscillates between using one gateway or the other, and the resulting average load distribution is far from optimal. The achieved goodput corresponds to a linear combination of the saturation goodputs of the best paths to each of the gateways and is significantly lower than the maximum. Hence, the system fails to take advantage of, and even suffers from, the presence of multiple gateways.

Applying RP$_\kappa$, the system may use both gateways simultaneously. When $\kappa = 1$, the achieved load distribution is close to optimal, which demonstrates that

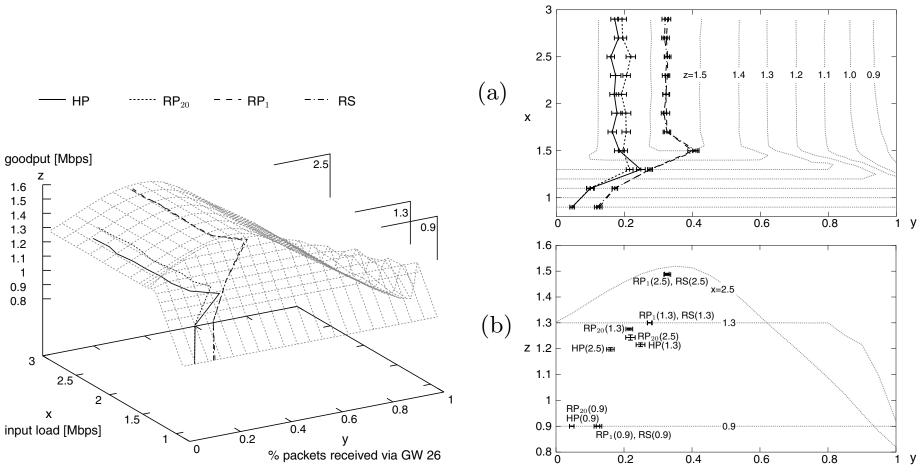


Fig. 4. Goodput and load distribution (source: 22; gateways: 1, 26; +-reachability)

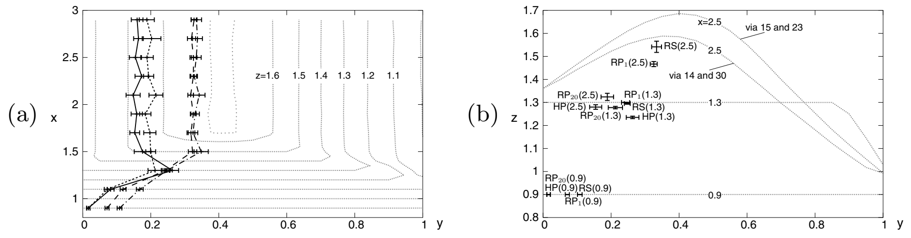


Fig. 5. Goodput and load distribution (source: 22; gateways: 1, 26; $*$-reachability)

ants do converge to the best solution. In the $*$-case, however, the distribution is not as close to the optimum as in the +-case. The reason is that OCEAN tends to reinforce, and thus converge to, a solution privileging most disjoint paths[4], here through 14 and 30. In both cases, the achieved goodput is higher than what it would be using a single gateway. In the +-case, it is close to the maximum. In the $*$-case, it is significantly lower than the maximum (also considering the maximum using paths through 14 and 30) due to interferences, similarly to what is observed when a single gateway is available. Finally, increasing κ increases the weight put on the best paths and, hence, here, towards the closest gateway, and results in a distribution farther from the optimum and a reduced goodput.

[4] The results obtained using the unicast version of CEAS, not included here due to space limitations, show that the system is able to converge to the optimal balance. However, this comes at the cost of significantly increased overhead and convergence time, which outweigh the benefits of achieving optimal load distribution.

3.2 Random Proportional Forwarding Using Selected Trails (RS)

Based on the above observations, a novel hybrid forwarding policy (RS) capturing the best of both existing policies is introduced. The idea is to forward packets on at most one path towards a given gateway to limit the intra-flow interferences, while still enabling the system to benefit from the presence of multiple gateways when there exist disjoint, or partly disjoint, paths to the gateways. For that purpose, ants record the gateway they sample and mark the pheromone trail accordingly. When forwarding data packets, these markings are used at each node to select only the neighbours with the highest pheromone trail values towards each of the locally known gateways. Packets are then forwarded stochastically amongst the selected neighbours proportionally to their relative pheromone levels. Note that the local forwarding choice does not presume anything about the ultimate gateway that will be used. More than which gateway, it is the forwarding direction that matters.

The probability of forwarding a data packet to node i at node v is

$$p_{t_v,vi} = \mathbf{I}(i \in \mathcal{G}_v) \cdot \frac{\tau_{t_v,vi}}{\sum_{j \in \mathcal{G}_v} \tau_{t_v,vj}}, \ \forall i \in \mathcal{N}_v \ , \tag{3}$$

where $\mathbf{I}$ denotes the indicator function of $\mathcal{G}_v$ and $\mathcal{G}_v$ is the set of neighbours of v such that $i \in \mathcal{G}_v \Leftrightarrow \tau_{t_v,vi} \geqslant \tau_{t_v,vj}, \ \forall j \in \mathcal{N}_v^{g_i}$, where $\mathcal{N}_v^g \subseteq \mathcal{N}_v$ is the set of neighbours of v which pheromone trail at v has last been reinforced by an ant backtracking from gateway g, and g_i is the gateway-marking of the pheromone trail towards i at v.

When only one gateway is available, RS is equivalent to RP$_\infty$, i.e. often HP in practice, and thus maximizes the achievable goodput. Figures 4 and 5 show the results obtained when multiple gateways are available re-using the same evaluation settings as in Sect. 3.1. The load distribution achieved applying RS is similar to what is achieved applying RP$_1$ and close to optimal. The achieved goodput is higher and also close to the maximum, showing that RS effectively reduces intra-flow interferences.

3.3 Convergence Time and Stability

The previous sections concentrate on achieving the maximum goodput and the optimal load distribution. In the context of path management in dynamic telecommunication networks, finding a good and stable solution in short time is at least as important as finding the optimal solution, and there is often a trade-off between the quality of the solution and the time to find it. How the different forwarding policies perform in these respects is briefly discussed below by looking at how the system adapts to changes. Note that the actual convergence speed scales with the ant generation rate and depends on the configuration of the system. These effects are not studied here. The focus is on comparing the different forwarding policies.

The change considered here is a second MR (node 0) starting to send data traffic while the network is operating in a steady state. The evaluation settings

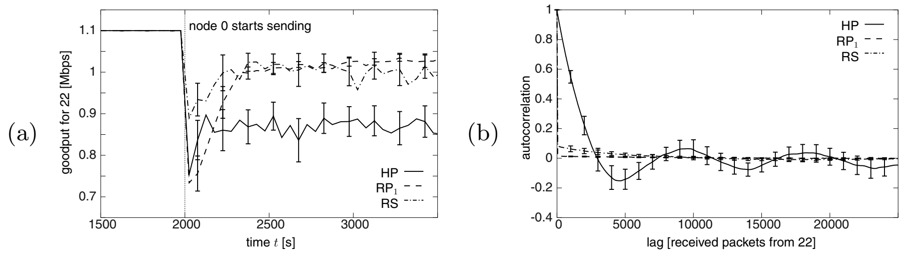


Fig. 6. Goodput and autocorrelation (sources: 22, 0; gateways: 1, 26; $\ast$-reachability)

are otherwise the same as in Sect. 3.1. Two gateways available (nodes 1 and 26). The $\ast$-reachability pattern is used. Both sources send CBR packets at 1.1 Mbps. When only 22 is sending, all the packets go through 1. When both MRs are sending, the maximal goodput for 22 is obtained by sending part of the traffic through 26. Figure 6(a) shows how the goodput for data generated at 22 varies in time. The fastest convergence is achieved applying HP. However, the achieved goodput is also the lowest because only the best path is used at a time and the system oscillates between using one gateway or the other. Fig. 6(b) shows the autocorrelation of the gateway used by data packets generated at 22 and received at the destination between $t = 3000$ [s] and 3500 [s], and characterizes the route flapping when applying HP. Note that the oscillation period is related to the link quality measurement period, here 20 seconds. Applying RP_1, the convergence is slower because the load is in a first time distributed on multiple paths, but all towards 1. Once the system has converged, the load is sent through both gateways resulting in a higher and much more stable goodput. Finally, applying RS, the system achieves a goodput similar to what is obtained applying RP_1, while improving the convergence time by avoiding to spread the load on paths all leading to the same congested area. This comes however at the cost of a weaker stability caused by oscillations in the paths to each of the gateways.

4 Concluding Remarks

In this paper, a novel ant-based multipath forwarding policy for WMNs is introduced. The main idea is to reduce intra-flow interferences by using at most one path to a given gateway at a time. In the context of WMNs, and as opposed to in MANETs, spending resources (time and bandwidth) for proactive optimization makes sense, and ant-based techniques are attractive candidates. One of the main trends to address routing issues in WMNs has been the development of complex multi-dimensional cost metrics, while still relying on traditional routing mechanisms. The approach followed here, on the contrary, is to rely on the (ant-based) routing system instead. Results are promising and indicate that adaptive and efficient load distribution can be achieved. Future work includes testing on a wider range of scenarios and comparing with other load distribution schemes.

References

1. Akyildiz, I., Wang, X., Wang, W.: Wireless mesh networks: a survey. Computer Networks 47(4), 445–487 (2005)
2. Baras, J.S., Mehta, H.: A probabilistic emergent routing algorithm for mobile ad hoc networks. In: 1st International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt (2003)
3. Campista, M.E.M., et al.: Routing metrics and protocols for wireless mesh networks. IEEE Network 22(1), 6–12 (2008)
4. Cetinkaya, C.: Improving the efficiency of multipath traffic via opportunistic traffic scheduling. Computer Networks 51(8), 2181–2197 (2007)
5. De Couto, D., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. In: 9th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom (2003)
6. Di Caro, G.A., Ducatelle, F., Gambardella, L.M.: AntHocNet: An ant-based hybrid routing algorithm for mobile ad hoc networks. In: Yao, X., et al. (eds.) PPSN VIII 2004. LNCS, vol. 3242, pp. 461–470. Springer, Heidelberg (2004)
7. Di Caro, G.A.: Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks. Ph.D. thesis, Université libre de Bruxelles, Belgium (2004)
8. Di Caro, G.A., Dorigo, M.: AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
9. Dorigo, M., Di Caro, G.A., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial Life 5(2), 137–172 (1999)
10. Ducatelle, F., Di Caro, G.A., Gambardella, L.M.: An analysis of the different components of the antHocNet routing algorithm. In: Dorigo, M., et al. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 37–48. Springer, Heidelberg (2006)
11. Ducatelle, F., Di Caro, G.A., Gambardella, L.M.: Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. Swarm Intelligence 4(3), 173–198 (2010)
12. Ducatelle, F.: Adaptive Routing in Ad Hoc Wireless Multi-hop Networks. Ph.D. thesis, University of Lugano, Switzerland (2007)
13. Heegaard, P.E., Wittner, O.J.: Overhead reduction in distributed path management system. Computer Networks 54(6), 1019–1041 (2010)
14. Huang, C.F., Lee, H.W., Tseng, Y.C.: A two-tier heterogeneous mobile ad hoc network architecture and its load-balance routing problem. Mobile Networks and Applications 9, 379–391 (2004)
15. Mueller, S., Tsang, R.P., Ghosal, D.: Multipath routing in mobile ad hoc networks: issues and challenges. In: Calzarossa, M.C., Gelenbe, E. (eds.) MASCOTS 2003. LNCS, vol. 2965, pp. 209–234. Springer, Heidelberg (2004)
16. Nandiraju, N., Nandiraju, D., Santhanam, L., He, B., Wang, J., Agrawal, D.: Wireless mesh networks: Current challenges and future directions of web-in-the-sky. IEEE Wireless Communications 14(4), 79–89 (2007)
17. Paquereau, L., Helvik, B.E.: Opportunistic ant-based path management for wireless mesh networks. In: Dorigo, M., et al. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 480–487. Springer, Heidelberg (2010)
18. Raghunathan, V., Kumar, P.R.: Wardrop routing in wireless networks. IEEE Transactions on Mobile Computing 8(5), 636–652 (2009)
19. Tarique, M., Tepe, K.E., Adibi, S., Erfani, S.: Survey of multipath routing protocols for mobile ad hoc networks. Journal of Network and Computer Applications 32, 1125–1143 (2009)

A Multiobjective Gravitational Search Algorithm Applied to the Static Routing and Wavelength Assignment Problem

Álvaro Rubio-Largo, Miguel A. Vega-Rodríguez, Juan A. Gómez-Pulido, and Juan M. Sánchez-Pérez

Department of Technologies of Computers and Communications, University of Extremadura, Polytechnic School, Cáceres, 10003 Spain
{arl,mavega,jangomez,sanperez}@unex.es

Abstract. One of the most favorable technology for exploiting the huge bandwidth of optical networks is known as Wavelength Division Multiplexing (WDM). Given a set of demands, the problem of setting up all connection requests is known as Routing and Wavelength Assignment (RWA) problem. In this work, we suggest the use of computational swarm intelligent for solving the RWA problem. A new heuristic based on the law of gravity and mass interactions (Gravitational Search Algorithm, GSA) is chosen for this purpose, but adapted to a multiobjective context (MO-GSA). To test the performance of the MO-GSA, we have used a real-world topology, the Nippon Telegraph and Telephone (NTT, Japan) network and six sets of demands. After performing several comparisons with other approaches published in the literature, we can conclude that this algorithm outperforms the results obtained by other authors.

Keywords: Gravitational Search Algorithm, Routing and Wave-length Assignment, Wavelength Division Multiplexing, Multiobjective Optimization.

1 Introduction

Currently, the number of users that use data networks has grown exponentially and their needs have evolved, requiring more bandwidth in their communication applications. The most favorable technology for exploiting the huge bandwidth of optical networks is known as Wavelength Division Multiplexing (WDM). This technique multiplies the available capacity of an optical fiber by adding new channels, each channel on a new wavelength of light. Everything points to the future of Internet will be based on WDM technology. Nowadays, several telecommunication companies in USA, Europe and Japan are testing and using several prototypes.

When it is necessary to interconnect a set of connection requests a problem comes up, this problem is known as *Routing and Wavelength Assignment* (*RWA*) problem. The optical connections carried end-to-end from a source node to a

© Springer-Verlag Berlin Heidelberg 2011

destination node over a wavelength on each intermediate link are known as *lightpaths*) [11]. Depending on the demands, we can distinguish two varieties of the RWA problem: *Static-RWA* and *Dynamic-RWA*. In Static-RWA the set of demands is known in advance, so the goal is to establish all demands minimizing the network resources. On the other hand, in the Dynamic-RWA problem, a lightpath is set up for each single request and after a finite period of time is erased. In this paper, we have focused on solving the Static RWA problem due to it is the most common one. Nowadays, WANs (Wide Area Networks) have static RWA because they are oriented to precontracted services [4].

In this work, we use a new heuristic based on the law of gravity and mass interactions, the Gravitational Search Algorithm (GSA) ([7]), but adapted to a multiobjective context (MO-GSA). To test the performance of the MO-GSA, we have used a real-world topology, the *Nippon Telegraph and Telephone* (NTT, Japan) network and six sets of demands. After performing several comparisons with other approaches published in the literature, we can conclude that this algorithm outperforms the results obtained by other authors.

The organization of the paper is as follows. In Section 2, we present the mathematical formulation of the Static-RWA problem. In Section 3, the proposed metaheuristic is outlined. Finally, several comparisons with other approaches published in the literature are shown in Section 4, and the relevant conclusions are drawn in Section 5.

2 Static RWA Problem Formulation

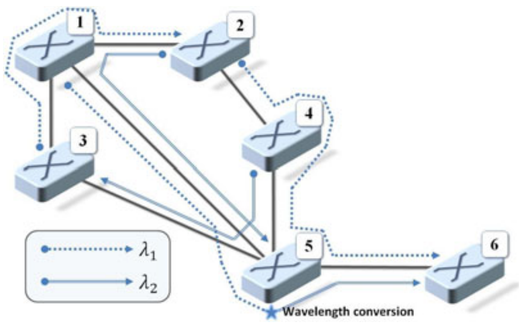
In this paper, an optical network is modeled as a direct graph $G = (V, E, C)$, where V is the set of nodes, E is the set of links between nodes and C is the set of available wavelengths for each optical link in E.

- $(i, j) \in E$: Optical link from node i to node j.
- $c_{ij} \in C$: Number of channels or different wavelengths at link (i, j).
- $u = (s_u, d_u)$: Unicast request u with source node s_u and destination node d_u, where $s_u, d_u \in V$.
- U : Set of demands, where $U = \{\ u \mid u \text{ is an unicast request}\}$.
- $|U|$: Cardinality of U.
- $u^{\lambda}_{i,j}$: Wavelength (λ) assigned to the unicast request u at link (i, j).
- l_u : Lightpath or set of links between a source node s_u and a destination node d_u; with the corresponding wavelength assignment in each link (i, j).
- L_u : Solution of the RWA problem considering the set of U requests.

Notice that $L_u = \{l_u | l_u$ is the set of links with their corresponding wavelength assignment$\}$. Using the above definitions, the RWA problem may be stated as a Multiobjective Optimization Problem (MOOP) [2], searching the best solution L_u that simultaneously minimizes the following two objective functions:

1. *Number of Hops* (y_1):

$$y_1 = \textstyle\sum_{u \in U} \sum_{(i,j) \in l_u} \Phi_j where \begin{Bmatrix} \Phi_j = 1 & \text{if} & (i,j) \in l_u \\ \Phi_j = 0 & \text{if} & otherwise \end{Bmatrix} \tag{1}$$



U	Lightpath	y_1	y_2
(2,6)	2 - λ_1 - 4 - λ_1 - 5 - λ_1 - 6	3	0
(3,2)	3 - λ_1 - 1 - λ_1 - 2	2	0
(2,5)	2 - λ_2 - 1 - λ_2 - 5	2	0
(4,3)	4 - λ_2 - 5 - λ_2 - 3	2	0
(1,6)	1 - λ_1 - 5 - λ_2 - 6	2	1
		y_1=11	y_2=1

Fig. 1. A simple example of the Static Routing and Wavelength Assignment problem

2. *Number of wavelength conversions* (y_2):

$$y_2 = \sum_{u \in U} \sum_{j \in V} \varphi_j where \begin{Bmatrix} \varphi_j = 1 & \text{if} & j \in V switches\ \lambda \\ \varphi_j = 0 & \text{if} & otherwise \end{Bmatrix} \tag{2}$$

Furthermore, we have to fulfill *the wavelength conflict constraint*: Two different unicast transmissions must be allocated with different wavelengths when they are transmitted through the same optical link (i, j). We use these objective functions with the aim of comparing with other authors ([1] and [5]).

An example helps to understand the problem formulation and the objective functions of the Static-RWA problem. Given the optical network topology of Figure 1, suppose the following set of demands: (2,6), (3,2), (2,5), (4,3) and (1,6); and two different wavelengths at link ($c_{ij} = 2$). As we can see in Figure 1, the following demands (2,6), (3,2), (2,5) and (4,3) do not present any wavelength conversion; however, the demand (1,6) presents one wavelength conversion in node 5. Furthermore, we present all necessary calculations to obtain the value of the two objective functions, *Number of Hops* (y_1) and *Number of Wavelength Conversions* (y_2). The solution could not be the best one; this example only tries to help to understand the problem formulation and the objective functions.

3 Multiobjective Gravitational Search Algorithm

The aim of this section is to present the algorithm used in this work for solving the Static-RWA problem. In first place, we present a brief description about the individuals, and subsequently, we describe the innovative multiobjective version of the Gravitational Search Algorithm (MO-GSA).

3.1 Representation of the Individuals

In this paper, the individuals of the algorithms have been designed as is shown in Figure 2. For each demand of the set, we execute a modified version of the

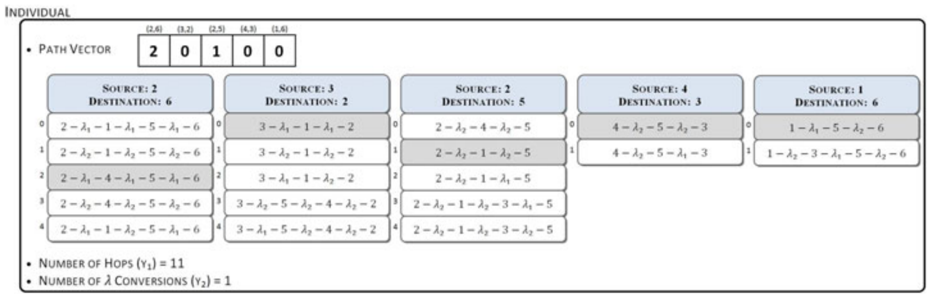

INDIVIDUAL
• PATH VECTOR
(2,6) (3,2) (2,5) (4,3) (1,6)
2 0 1 0 0
SOURCE: 2 DESTINATION: 6
0 $2-\lambda_1-1-\lambda_1-5-\lambda_1-6$
1 $2-\lambda_2-1-\lambda_2-5-\lambda_2-6$
2 $2-\lambda_1-4-\lambda_1-5-\lambda_1-6$
3 $2-\lambda_2-4-\lambda_2-5-\lambda_2-6$
4 $2-\lambda_1-1-\lambda_2-5-\lambda_1-6$
SOURCE: 3 DESTINATION: 2
0 $3-\lambda_1-1-\lambda_1-2$
1 $3-\lambda_2-1-\lambda_2-2$
2 $3-\lambda_1-1-\lambda_2-2$
3 $3-\lambda_2-5-\lambda_2-4-\lambda_2-2$
4 $3-\lambda_1-5-\lambda_2-4-\lambda_2-2$
SOURCE: 2 DESTINATION: 5
0 $2-\lambda_2-4-\lambda_2-5$
1 $2-\lambda_2-1-\lambda_2-5$
2 $2-\lambda_2-1-\lambda_1-5$
3 $2-\lambda_2-1-\lambda_2-3-\lambda_1-5$
4 $2-\lambda_2-1-\lambda_2-3-\lambda_2-5$
SOURCE: 4 DESTINATION: 3
0 $4-\lambda_2-5-\lambda_2-3$
1 $4-\lambda_2-5-\lambda_1-3$
SOURCE: 1 DESTINATION: 6
0 $1-\lambda_1-5-\lambda_2-6$
1 $1-\lambda_2-3-\lambda_1-5-\lambda_2-6$
• NUMBER OF HOPS (Y_1) = 11
• NUMBER OF λ CONVERSIONS (Y_2) = 1

Fig. 2. Individual that provides the solution: y_1=11, y_2=1. It is obtained executing the modified version of the Yen´s algorithm with k_{max}=5 for every demand of the set: {(2,6), (3,2), (2,5), (4,3) and (1,6)}.

Yen´s algorithm [12]. In our version, we have introduced an heuristic to assign the wavelengths. This heuristic tries to assign always the same wavelength (λ) which was previously assigned (previous hop).

If it is not possible, or it is the first assignation, the heuristic choices the first free λ. Using this modified version of the Yen´s algorithm, we create a list with as maximum k_{max} possible routes (including wavelengths) and we select one of them. This selection is stored in a vector, as it is shown in Figure 2. This procedure is repeated for every demand of the set.

3.2 Description for MO-GSA

The Gravitational Search Algorithm (GSA) is a population based algorithm created by Rashedi et al. [7]. This new optimization algorithm is based on the law of gravity and mass interactions. In this proposal, the searcher agents (individuals) are a collection of masses which interact with each other based on the Newtonian gravity laws of motion.

Since the RWA problem is a MOOP, we have to adapt the GSA to a multiobjective context. We have used concepts of well-known multiobjective algorithms such as Non-Dominated Sorting Genetic Algorithm (NSGA-II, [3]) and Pareto Archived Evolution Strategy (PAES, [6]). From the NSGA-II we have taken the *FastNonDominatedSort* which sorts the population into different nondomination levels such as the typical NSGA-II. From PAES, we have taken the idea of a nondominated solution archive (NDS archive), using the same acceptance function.

In Algorithm 1, we present the pseudocode for our multiobjective version of the Gravitational Search Algorithm (MO-GSA). In this particular problem, Static-RWA problem, we have considered $|U|$ (number of demands), as the number of dimensions, hence, the positions to update are the cells of the corresponding path vector associated to an individual (see Figure 2) .

The algorithm starts with the random generation of N individuals (line 4 in Algorithm 1) through running our modified version of Yen´s algorithm for each

Algorithm 1. Pseudocode for MO-GSA

```
1.  NDSarchive ⇐ ∅
2.  $K_{best}$ ⇐ $N$
3.  /* Generate Initial Population $P = \{X_1, X_2, ..., X_N\}$ */
4.  $P$ ⇐ generateRandomPopulation($N$)
5.  while not time-limit do
6.      /* Evaluate the fitness for each individual */
7.      $P$ ⇐ fastNonDominatedSort($P$)
8.      $P$ ⇐ CrowdingDistanceAssignment($P$)
9.      $P$ ⇐ calculateMOFitness($P$)
10.     /* Update $G$, $MOFitness_{best}$ and $MOFitness_{worst}$ of the population $P$ */
11.     $G \Leftarrow G_0 e^{-\alpha \frac{t}{T}}$
12.     $MOFitness_{best} \Leftarrow \max_{i \in \{1,...,N\}} X_i$.MOFitness
13.     $MOFitness_{worst} \Leftarrow \min_{i \in \{1,...,N\}} X_i$.MOFitness
14.     /* Calculate mass and acceleration for each individual */
15.     for i=1 to $N$ do
16.         $X_i.q \Leftarrow \frac{X_i.MOFitness - MOFitness_{worst}}{MOFitness_{best} - MOFitness_{worst}}$
17.     end for
18.     for i=1 to $N$ do
19.         $X_i$.mass ⇐ $\frac{X_i.q}{\sum_{j=1}^{N} X_j.q}$
20.     end for
21.     for d=1 to $|U|$ do
22.         for i=1 to $N$ do
23.             for j=1 to $K_{best}$ do
24.                 $R_{ij} \Leftarrow \|X_i, X_j\|_2$ // Euclidean Distance between $X_i$ and $X_j$
25.                 $X_i.F_j^d \Leftarrow G \times \frac{X_i.mass \times X_j.mass}{R_{ij}+\varepsilon} \times (X_j.pathVector^d - X_i.pathVector^d)$
26.             end for
27.             $X_i.force^d \Leftarrow \sum_{j \in K_{best}, j \neq i}^{N} rand[0,1] \times X_i.F_j^d$
28.             $X_i.acceleration^d \Leftarrow \frac{X_i.force^d}{X_i.mass}$
29.         end for
30.     end for
31.     /* Update velocity and position of every demand of each individual */
32.     for d=1 to $|U|$ do
33.         for i=1 to $N$ do
34.             $X_i.velocity^d \Leftarrow rand[0,1] \times X_i.velocity^d + X_i.acceleration^d$
35.             $X_i.pathVector^d \Leftarrow X_i.pathVector^d + X_i.velocity^d$
36.         end for
37.     end for
38.     NDSarchive ⇐ updateNDSarchive($P$)
39.     $K_{best}$ ⇐ Decrease($K_b est$)
40.     /* Check if occur stagnation */
41.     if stagnation then
42.         $P$ ⇐ Mutation($P$)
43.     end if
44. end while
```

demand of the set, generating for each individual a path vector and storing them in population P. Each isolated individual (during the random generation of P) is added to the NDS archive.

After generating the pool of agents, we classify the population into different pareto fronts, by ranking the individuals (from 1 to N) and sorting them by rank (line 7). Furthermore, we calculate the crowding distance of each individual (line 8) with the aim of calculating the MOFitness (line 9). In equation 3, the calculation to obtain the MOFitness of an individual is shown which needs to be maximum. For example, if we compare the worst individual with rank 1 and lowest value of crowding distance (≈ 0) and the best individual with rank 2 and highest crowding distance (∞), they present a MOFitness value of 1/3 and 1/4 respectively.

$$MOFitness(X_i) = (2^{X_i.rank} + \frac{1}{1 + X_i.crowding\_distance})^{-1} \qquad (3)$$

Subsequently, we update the gravitational constant (G), the best and the worst MOFitness. To update G, we have used the same equation proposed in [7] by the authors (see line 11), where t is the actual generation, T is the total number of generations and α is used to measure the reduction of G. Since the MOFitness must be maximum, we have to update the best and the worst MOFitness using the best (highest value of MOFitness) individual and the worst individual (lowest value of MOFitness).

Next, we calculate the mass and acceleration of each agent of the population. The masses are simply calculated by the MOFitness evaluation. A heavier mass means a more efficient agent (lines 15-20). In first place, we have to calculate, for each dimension, the force acting on each individual X_i from all other individuals that belong to the set K_{best}, by using the equation presented in line 25. The set K_{best} is suggested by the authors in [7], to improve the performance of the heuristic. In that way, the agents will be affected only by the k best individuals of the population. The value of K_{best} is set to N (population size) at the beginning, and it is decreased linearly (line 39) as time goes by. To give a stochastic characteristic to the GSA algorithm, the total force (line 27) that acts on X_i in a dimension d must be a randomly (in the interval [0,1]) weighted sum of d^{th} components of the forces exerted from the K_{best} individuals. By using the total force in a dimension d and the mass associated to an individual, we calculate the acceleration suffered by the individual in the dimension d (line 28).

Once calculated the mass and acceleration of each individual, we have to calculate the corresponding velocity (in each dimension) with the aim of improving the position of the individual. The velocity of an individual is considered as a fraction of its current velocity added to its acceleration. In lines 32-37, we show the appropriate equations to update the velocity and position of each individual of the population.

In line 38, we add to the NDS archive the whole population, in that way, we are able to return a pareto front when the algorithm finishes its execution. We have added an improvement to the original GSA, a mutation in case of stagnation (lines 41-43). The mutation is applied only in case of several generations without changes in the NDS archive. Finally, the MO-GSA restarts a new generation until the time limit for the experiment is expired (line 5).

4 Empirical Results

In this section we show several comparisons with other approaches published in the literature.

Two novel multiobjective evolutionary algorithms for solving the Static RWA problem have been published recently: the classical Differential Evolution, but adapted to multiobjective context with the Pareto Tournaments concept (DEPT, [8]) and a multiobjective version of the Variable Neighborhood Search algorithm (MO-VNS, [9]). In this work, we have also applied the standard multiobjective Non-Dominated Sorting Genetic Algorithm (NSGA-II, [3]) to the Static RWA problem with the aim of comparing with MO-GSA.

Other authors has also tackled the Static RWA problem. In [1], the authors make a comparison between typical heuristics in telecommunication field

Table 1. Runtimes, reference points (r_{min} and r_{max}) to calculate the hypervolume and Short-Names for each test instance

	$\lvert U \rvert$	c_{ij}	Runtime (s)	r_{min}	r_{max}	Short Name
NTT	10	10	6	(0, 0)	(220, 20)	NTT1
		20	65	(0, 0)	(530, 70)	NTT2
		40	110	(0, 0)	(790, 190)	NTT3

	$\lvert U \rvert$	c_{ij}	Runtime (s)	r_{min}	r_{max}	Short Name
NTT	8	10	6	(0, 0)	(230, 20)	NTT4
		20	65	(0, 0)	(520, 110)	NTT5
		30	70	(0, 0)	(560, 80)	NTT6

Table 2. Best configuration found for: DEPT, MO-VNS, NSGA-II and MO-GSA

DEPT	
K-shortest-paths (k)	10
Population Size (N)	25
Crossover Probability (cr)	20%
Mutation Factor (f)	50%
Selection Schema (s)	Best/1/Bin

MO-VNS	
K-shortest-paths (k)	10

NSGA-II	
K-shortest-paths (k)	10
Population Size (N)	25
Crossover Probability (cr)	70%
Crossover Schema	SPX
Elitism Probability (e)	75%
Mutation Probability (f)	10%

MO-GSA	
K-shortest-paths (k)	10
Population Size (N)	25
Gravitational Constant (G_0)	100
Alpha (α)	20
Mutation Probability (f)	25%

and different varieties of Multiobjective Ant Colony Optimization algorithms (MOACO) for solving the Static RWA problem. The typical heuristics purposed in [1] are: 3SPFF (3-Shortest Path routing, First-Fit wavelength assignment), 3SPLU (3-Shortest Path routing, Least-Used wavelength assignment), 3SPMU (3-Shortest Path routing, Most-Used wavelength assignment), 3SPRR (3-Shortest Path routing, Random wavelength assignment), SPFF (Shortest Path Dijkstra routing, First-Fit wavelength assignment), SPLU (Shortest Path Dijkstra routing, Least-Used wavelength assignment), SPMU (Shortest Path Dijkstra routing, Most-Used wavelength assignment) and SPRR (Shortest Path Dijkstra routing, Random wavelength assignment). On the other hand, the different varieties of MOACOs in [5] and [1] are: BIANT (Bicriterion Ant), COMP (COMPETants), MOAQ (Multiple Objective Ant Q Algorithm), MOACS (Multi-Objective Ant Colony System), M3AS (Multiobjective Max-Min Ant System), MAS (Multiobjective Ant System), PACO (Pareto Ant Colony Optimization) and MOA (Multiobjective Omicron ACO). For further information about these approaches, refer to [1] and [5].

To carry out the comparisons, we have used a real-world network topology, the Nippon Telegraph and Telephone (NTT, Japan) network, and six sets of demands obtained from [10] (see Table 1). After performing a parameter tuning of each approach (DEPT, MO-VNS, NSGA-II and MO-GSA), the best configuration found for each one is shown in Table 2. The methodology used for adjusting the parameters of the algorithms is the following: 30 independent runs and an statistical analysis using ANOVA tests, in this way, we can say that the parameter tuning was statistically relevant.

However, in [1], the comparison is carried out using only the following sets of demands: NTT2, NTT3, NTT4 and NTT5. Therefore, when we compare with [1], we will use only these instances, comparing with the best results indicated in [1].

To compare the approaches, we have used two multiobjective metrics, *Hypervolume* [14] and *Coverage Relation* [13]. To calculate the hypervolume is necessary the use of two reference points, $r_{min}(x_{min}, y_{min})$ and $r_{max}(x_{max}, y_{max})$,

Table 3. Comparison among several algorithms published in the literature (DEPT [8], MO-VNS [9], NSGA-II [3], 3SPLU [1], MOA [5], BIANT [1] and MAS [5]) and our approach (MO-GSA) using the average *Hypervolume* of 30 independent runs

(a)

NTT1	
DEPT	69.55%
MO-VNS	69.55%
NSGA-II	69.55%
MO-GSA	69.55%

(b)

NTT2	
3SPLU	62.96%
MOA	56.01%
DEPT	69.43%
MO-VNS	68.81%
NSGA-II	69.81%
MO-GSA	69.73%

(c)

NTT3	
3SPLU	63.18%
BIANT	57.52%
DEPT	63.48%
MO-VNS	62.73%
NSGA-II	62.54%
MO-GSA	63.96%

(d)

NTT4	
3SPLU	70.87%
3SPRR	70.87%
SPLU	70.87%
SPRR	70.87%
M3AS	70.87%
MOA	70.87%
DEPT	70.87%
MO-VNS	70.87%
NSGA-II	70.87%
MO-GSA	70.87%

(e)

NTT5	
3SPLU	66.81%
MAS	64.79%
MOA	63.37%
DEPT	68.66%
MO-VNS	67.92%
NSGA-II	68.02%
MO-GSA	68.79%

(f)

NTT6	
DEPT	64.31%
MO-VNS	61.79%
NSGA-II	64.62%
MO-GSA	65.21%

where x is the number of hops (y_1) and y is the number of wavelength switchings (y_2). In Table 1 we show the different reference points for each data set. The r_{max} point for every data set was calculated from the experience.

Firstly, we compare the MO-GSA with the approaches mentioned above, using the hypervolume concept. As we can see in Table 3(a), all approaches obtained the same value of hypervolume, obtaining the optimal pareto front. In Table 3(b), the MO-GSA obtains slightly lower hypervolume than the standard NSGA-II, however, it obtains a significantly higher hypervolume than the approaches from other authors (3SPLU and MOA). In Table 3(c), we can see that the MO-GSA obtains better hypervolume than any other approach. As occurs before, in Table 3(d), all approaches obtains the same hypervolume. Finally, in Tables 3(e) and Table 3(f), we can notice that the MO-GSA overcomes the results of hypervolume achieved by any other heuristic. We can conclude that the MO-GSA provides, in almost all data sets, superior or equal value of hypervolume to the value obtained by the rest of approaches.

Secondly, we present a direct comparison (Coverage Relation) of the outcomes achieved by the algorithms presented above. This metric measures the fraction of non-dominated solutions evolved by an algorithm B, which are covered by the nondominated points achieved by an algorithm A in average. Once performed the first comparison, we decide to make the second comparison using almost all data sets, except NTT1 and NTT4, due to the fact that all approaches have obtained the same pareto front. In Table 4(a), we can see that the pareto front obtained by MO-GSA dominates the fronts obtained by the best MOACO (MOA) and by the best typical heuristic (3SPLU). Furthermore, its front covers the 100%, 80% and 80% the fronts obtained by DEPT, MO-VNS and NSGA-II, respectively. In Table 4(b), the DEPT and NSGA-II covers the 33.33% of the surface of MO-GSA, by contrast, the MO-GSA is able to covers completely the surface of BIANT, 3SPLU and MO-VNS and 50% and 75% of DEPT and NSGA-II. In Table 4(c), we can notice that the MO-GSA front is better or equal than all the other fronts obtained by the other algorithms. We can see that the front obtained by 3SPLU does not dominate the front obtained by the MO-GSA, and vice versa. Finally, in Table 4(d) we show a Coverage Relation among the DEPT,

Table 4. Comparison among several algorithms published in the literature (DEPT [8], MO-VNS [9], NSGA-II [3], 3SPLU [1], MOA [5], BIANT [1] and MAS [5]) and our approach (MO-GSA) using the *Coverage Relation* metric ($A \succeq B$)

(a)

NTT2					
A	MOA	3SPLU	DEPT	MO-VNS	NSGA-II
B			MO-GSA		
	0%	0%	0%	33.33%	0%
A			MO-GSA		
B	MOA	3SPLU	DEPT	MO-VNS	NSGA-II
	100%	100%	100%	80%	80%

(b)

NTT3					
A	BIANT	3SPLU	DEPT	MO-VNS	NSGA-II
B			MO-GSA		
	0%	0%	33.33%	0%	33.33%
A			MO-GSA		
B	BIANT	3SPLU	DEPT	MO-VNS	NSGA-II
	100%	100%	50%	100%	75%

(c)

NTT5						
A	MAS	MOA	3SPLU	DEPT	MO-VNS	NSGA-II
B				MO-GSA		
	0%	0%	0%	0%	100%	0%
A				MO-GSA		
B	MAS	MOA	3SPLU	DEPT	MO-VNS	NSGA-II
	62.50%	50%	0%	100%	100%	100%

(d)

NTT6			
A	DEPT	MO-VNS	NSGA-II
B		MO-GSA	
	50%	16.67%	50%
A		MO-GSA	
B	DEPT	MO-VNS	NSGA-II
	80%	100%	80%

MO-VNS, NSGA-II and MO-GSA, we can check that the pareto front obtained by MO-GSA covers a 80%, 100% and 80% of the surface of the front obtained by DEPT, MO-VNS and NSGA-II, respectively.

To sum up, after performing a exhaustive comparison among the best typical heuristics proposed in [1], the best MOACOs proposed in [1] and [5], DEPT, MO-VNS and NSGA-II, we can say that the MO-GSA obtains very promising results. It obtains better results than almost twenty heuristics.

5 Conclusions and Future Work

In this paper, a real-world optical networking problem has been solved by using an innovative multiobjective version of the Gravitational Search Algorithm (MO-GSA). A real-world topology has been used in the experimental section, the Nippon Telegraph and Telephone (NTT, Japan) network and six sets of demands. After performing several comparisons with several approaches published in the literature, we can conclude saying that the MO-GSA outperforms the results obtained by any other approach in almost all data sets. As future work, we intend to apply this innovative multiobjective algorithm to the Dynamic RWA problem. Furthermore, it would be interesting to develop different swarm intelligent algorithms (as those based on bee colonies) with the aim of making comparisons with MO-GSA.

Acknowledgements. Álvaro Rubio-Largo is supported by the research grant PRE09010 from Junta de Extremadura. This work has been partially funded by the Spanish Ministry of Education and Science and ERDF (the European Regional Development Fund), under contract TIN2008-06491-C04-04 (the M* project).

References

1. Arteta, A., Barán, B., Pinto, D.: Routing and Wavelength Assignment over WDM Optical Networks: a comparison between MOACOs and classical approaches. In: LANC 2007: Proceedings of the 4th International IFIP/ACM Latin American Conference on Networking, pp. 53–63. ACM, New York (2007)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2000)
4. Hamad, A.M., Kamal, A.E.: A survey of multicasting protocols for broadcast-and-select single-hop networks. IEEE Network 16, 36–48 (2002)
5. Insfrán, C., Pinto, D., Barán, B.: Diseño de Topologías Virtuales en Redes Ópticas. Un enfoque basado en Colonia de Hormigas. In: XXXII Latin-American Conference on Informatics, CLEI 2006, vol. 8, pp. 173–195 (2006)
6. Knowles, J., Corne, D.: The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 1, p. 105 (1999)
7. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: A Gravitational Search Algorithm. Information Sciences 179(13), 2232–2248 (2009); Special Section on High Order Fuzzy Sets
8. Rubio-Largo, A., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: A Differential Evolution with Pareto Tournaments for solving the Routing and Wavelength Assignment Problem in WDM Networks. In: Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC 2010), vol. 10, pp. 129–136 (2010)
9. Rubio-Largo, A., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Solving the routing and wavelength assignment problem in WDM networks by using a multiobjective variable neighborhood search algorithm. In: Corchado, E., Novais, P., Analide, C., Sedano, J. (eds.) SOCO 2010. AISC, vol. 73, pp. 47–54. Springer, Heidelberg (2010)
10. Schaerer, M., Barán, B.: A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. In: IASTED International Conference on Applied Informatics, pp. 97–102 (2003)
11. Siva Ram Murthy, C., Gurusamy, M.: WDM Optical Networks - Concepts, Design and Algorithms. Prentice Hall PTR, Englewood Cliffs (2002)
12. Yen, J.Y.: Finding the K Shortest loopless paths in a Network. Manage Sci. 17(11), 712–716 (2003)
13. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8, 173–195 (2000)
14. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study, pp. 292–301. Springer, Heidelberg (1998)

A Population Based Incremental Learning for Delay Constrained Network Coding Resource Minimization

Huanlai Xing and Rong Qu

The Automated Scheduling, Optimisation and Planning (ASAP) Group
School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK
{hxx,rxq}@cs.nott.ac.uk

Abstract. In network coding based multicast, coding operations are expected to be minimized as they not only incur additional computational cost at corresponding nodes in network but also increase data transmission delay. On the other hand, delay constraint must be concerned particularly in delay sensitive applications, e.g. video conferencing. In this paper, we study the problem of minimizing the amount of coding operations required while meeting the end-to-end delay constraint in network coding based multicast. A population based incremental learning (PBIL) algorithm is developed, where a group of best so far individuals, rather than a single one, is maintained and used to update the probability vector, which enhances the global search capability of the algorithm. Simulation results demonstrate the effectiveness of our PBIL.

Keywords: multicast; network coding; population based incremental learning.

1 Introduction

Network coding has been attracting an increasing research attention since 2000 [1]. Instead of simply replicating and forwarding data packets at the network layer, network coding allows any intermediate node, if necessary, to perform mathematical operations to recombine data packets received from different incoming links. By doing so, the maximized multicast throughput is always obtained [2].

In most of the previous research on network coding, coding is performed at all coding-possible nodes without concerning issues raised in real life applications. One such issue is that, to obtain an expected multicast throughput, coding may only be necessary at a subset of coding-possible nodes [3,4,5]. As they consume public resources, e.g. buffer and computational resources, coding operations should be minimized to leave more public resources for other network applications. Another issue in real time applications is that transmission delays, especially in delay sensitive applications, e.g. video conferencing and distributed game, should be bounded. It is therefore important to minimize coding operations while meeting the end-to-end delay requirement. However, such problem has not drawn enough research attention.

Several algorithms have been proposed to minimize coding resources, however, without concerning the delay constraint. In [6] and [7], original topologies were decomposed and transformed into secondary graphs, based on which greedy methods

© Springer-Verlag Berlin Heidelberg 2011

were developed to reduce the amount of coding operations. In [8], linear programming formulations based on a model allowing continuous flows were proposed to optimize various resources used for network coding. In addition, Kim *et al* investigated centralized and distributed genetic algorithms (GAs) with problem-specific operators to minimize the required network coding resources [3,4,5].

Population based incremental learning (PBIL), a combination of evolutionary algorithm and competitive learning, was first introduced in 1994 [9]. Without using crossover and mutation, PBIL retains the stochastic search nature of GA by simply maintaining a probability vector. Since its introduction, PBIL has shown to be very successful on numerous benchmark and real-world problems [10].

In this paper, we propose the first PBIL algorithm for the problem of minimizing network coding resources with the delay bound in network coding based multicast. We put forward a new probability vector update scheme based on statistical information of a set of best so far individuals. In addition, we observed that the use of the all-one vector in the initialization of PBIL greatly helps to guide the search towards promising solutions. Simulation results demonstrated that our PBIL is highly effective for the problem concerned.

2 Problem Description

A communication network can be modeled as a directed graph $G = (V, E)$, where V and E denote the set of nodes and links, respectively [2]. We assume each link e has a unit capacity. A delay constrained single-source network coding based multicast scenario can be defined as a 5-tuple set (G, s, T, R, Ω), where the information needs to be transmitted at data rate R from the source node $s \in V$ to a set of sinks $T = \{t_1,\ldots,t_d\} \subset V$ in the graph G (V, E), satisfying a given end-to-end delay constraint Ω. Rate R is achievable if there is a transmission scheme that enables each sink t_k, $k = 1,\ldots,d$, to receive information at data rate R [4,5]. As each link has a unit capacity, a path from s to t_k thus has a unit capacity. If we manage to set up R link-disjoint paths $\{P_1(s, t_k),\ldots,P_R(s, t_k)\}$ from s to each sink $t_k \in T$, we make the data rate R achievable. We concern linear network coding which is sufficient for multicast [2].

In this paper, a subgraph is referred to as a *network coding based multicast subgraph* (NCM subgraph, denoted by $G_{NCM}(s, T)$) if there are R link-disjoint paths $P_i(s, t_k)$, $i = 1,\ldots,R$, from s to each sink t_k in this subgraph. We refer to a non-sink node with multiple incoming links as a *merging node* [4,5]. Only merging nodes and their outgoing coding links have the potential to serve as coding nodes and coding links, respectively [4,5]. To determine if an outgoing link of a merging node becomes a coding link, we only need to check if the information via this link is dependent on at least two incoming links of the merging node.

For a given multicast scenario, the number of coding links is more precise to indicate the total amount of coding operations [7]. We hereinafter investigate how to construct a NCM subgraph $G_{NCM}(s, T)$ with the minimal number of coding links while achieving the expected data rate and satisfying the end-to-end delay constraint Ω.

We define the following notations in our paper:

- x_{ij}: a variable associated with the j-th outgoing link of the i-th merging node, $i = 1,\dots,M$, $j = 1,\dots,Z_i$, where M is the total number of merging nodes and the i-th node has Z_i outgoing links. $x_{ij} = 1$ if the j-th outgoing link of the i-th node serves as a coding link; $x_{ij} = 0$ otherwise.
- $N_{cl}(G_{NCM}(s,T))$: the number of coding links in a NCM subgraph $G_{NCM}(s, T)$.
- $R(s, t_k)$: the achievable rate from s to t_k.
- R: the defined data rate (an integer) at which s expects to transmit information.
- $e(n_a, n_b)$: a directed link from node n_a to node n_b.
- $P_i(s, t_k)$: the i-th established path from s to t_k, $i = 1,\dots,R$ in $G_{NCM}(s, T)$.
- $W_i(s, t_k) = \{e \mid e \in P_i(s, t_k)\}$: the link set of $P_i(s, t_k)$.
- $P_i^{s\to t_k}(n_a, n_b)$: the subpath from node n_a to node n_b on the path $P_i(s, t_k)$. Along a path, we assume that node $n_{j\text{-}1}$ is the parent node of node n_j, $j = 2,\dots,p$, $n_1 = s$ and $n_p = t_k$.
- $D(\omega)$: the delay of the term ω. $D(\omega)$ represents the data processing delay of a node ω; or it is the propagation delay of a link ω. If ω is a path $P(n_i,n_j)$, $D(\omega)$ is the end-to-end delay from node n_i to node n_j; if ω is a NCM subgraph $G_{\mathrm{NCM}}(s,T)$, $D(\omega)$ denotes the maximal path delay in $G_{\mathrm{NCM}}(s,T)$.

Based on the above notations, we define the problem of delay constrained coding resource minimization as to minimize the number of coding links while achieving the desired multicast throughput and meeting the delay restriction, shown as follows:

$$\textit{Minimize}: \quad N_{cl}(G_{NCM}(s,T)) = \sum_{i=1}^{M}\sum_{j=1}^{Z_i} x_{ij} \tag{1}$$

$$\textit{Subject to}: \quad R(s, t_k) \geq R,\ \forall t_k \in T \tag{2}$$

$$\bigcap_{i=1}^{R} W_i(s,t_k) = \varnothing,\ \ \forall t_k \in T \tag{3}$$

$$D(G_{NCM}(s, T)) \leq \Omega \tag{4}$$

where,

$$D(G_{NCM}(s, T)) = \max\{D(P_i(s, t_k)) \mid i = 1, 2, \dots, R,\ \forall t_k \in T\} \tag{5}$$

Objective (1) defines our problem as to find a NCM subgraph with the minimal number of coding links; Constraint (2) defines that the practical achievable data rate from s to every sink must be at least R so that R paths can be constructed for each sink; Constraint (3) restricts that for each t_k the R constructed paths $P_i(s, t_k)$, must have no common link; Constraint (4) defines that the maximal delay of the constructed NCM subgraph cannot be greater than the pre-defined delay bound Ω; The delay of the NCM subgraph is defined in (5) as the maximal delay among all paths. Note that in this paper we only consider the transmission delay from s to t_k along each path $P_i(s, t_k)$. The decoding delay to obtain the original information in each sink is omitted.

Along a path $P_i(s, t_k)$, there are in general four types of nodes: the source, the sink, forwarding node(s) and coding node(s). If a node n_j is a sink or a forwarding node, we ignore its data processing delay, i.e. $D(n_j) = 0$. The delay of the path from s to the sink or the forwarding node is thus defined as follows:

$$D(P_i^{s\to t_k}(s,n_j)) = D(P_i^{s\to t_k}(s,n_{j-1})) + D(e(n_{j-1},n_j)) \quad (6)$$

If a number of data packets are required to be coded together at a coding node, coding operation cannot start until the arrival of the last data packet. If a node n_j is a coding node, we assume that among the $R\cdot d$ paths in $G_{NCM}(s, T)$ there are Y paths that pass through n_j. Obviously, along each of the Y paths there is a subpath from s to n_j. We denote the delays of the Y subpaths by $D_1(s, n_j)$, $D_2(s, n_j)$, …, $D_Y(s, n_j)$. The delay of the subpath from s to coding node n_j is defined as follows:

$$D(P_i^{s\to t_k}(s,n_j)) = \max\{D_r(s,n_j) \mid r = 1,...,Y\} + \Delta_c \quad (7)$$

where Δ_c is the time consumed by the coding operation. We assume any coding operation consumes the same time Δ_c.

3 The Proposed PBIL

As an estimation of distribution algorithm, PBIL maintains a real-valued probability vector which, when sampled, generates promising solutions with high probabilities. The procedure of the standard PBIL with elitism [9] is shown in Fig.1. In the probability vector $\boldsymbol{P}(t) = \{P_1^t, P_2^t,..., P_L^t\}$ at generation t, where L is the solution length, and P_i^t, $i = 1, 2, …, L$, is the probability of generating '1' at the i-th position of a solution. At each generation, $\boldsymbol{P}(t)$ is sampled to form a set $S(t)$ of N individuals (i.e. solutions) which are evaluated and assigned a fitness value using a given fitness function. Let α be the learning rate. The best so far individual $\boldsymbol{B}(t) = \{B_1^t, B_2^t,..., B_L^t\}$ is then selected to update $\boldsymbol{P}(t)$ as follows:

$$P_i^t = (1.0 - \alpha) \cdot P_i^t + \alpha \cdot B_i^t, \quad i = 1, 2, …, L \quad (8)$$

After the $\boldsymbol{P}(t)$ is updated, a bit-wise mutation operation may be adopted to maintain the diversity and avoid local optima [9]. We denote by pm and σ the mutation probability and the amount of mutation at each locus P_i^t, respectively. For each locus P_i^t, a uniformly distributed random number rnd_i in [0.0, 1.0] is generated. If $rnd_i < pm$, P_i^t is mutated by the following formula:

$$P_i^t = (1.0 - \sigma) \cdot P_i^t + rnd(\{0.0, 1.0\}) \cdot \sigma \quad (9)$$

where $rnd(\{0.0, 1.0\})$ is 0.0 or 1.0, randomly generated with a probability 0.5.

After mutation, a sampling set is generated by the new $\boldsymbol{P}(t)$. Step 6 to 11 is repeated until the termination condition is met. Along with the evolution, $\boldsymbol{P}(t)$ gradually converges to an explicit solution.

In this paper, we propose a new probability vector update scheme where a number of best so far individuals are adopted. In addition, an all-one vector is employed at the beginning of the algorithm to guide the search towards feasible solutions.

```
1)  Initialization
2)   Set $t = 0$;
3)    for $i = 1$ to $L$, do set $P_i^t = 0.5$
4)    Generate a sampling set $S(t)$ of $N$ individuals from $P(t)$
5)  repeat
6)    Set $t = t + 1$
7)    Evaluate the samples in $S(t\text{-}1)$
8)    Find the best individual $B(t)$ from $B(t\text{-}1) \cup S(t\text{-}1)$
9)    Update $P(t)$ by Eq.(8)
10)   Mutate $P(t)$ by Eq.(9)
11)   Generate set $S(t)$ by sampling $P(t)$
12) until termination condition is met
```

Fig. 1. Procedure of PBIL

```
1) Find the H best individuals from $S(t\text{-}1)$ and sort them in sequence $\{C_1, C_2,\dots, C_H\}$, where $f(C_1) \le f(C_2) \le \dots \le f(C_H)$
2) for $i = 1$ to H do
3)     Find the worst individual $B_{MAX}$ in the set of best so far individuals $S_{BSF}$, where $f(B_{MAX}) = \max\{ f(B_1), f(B_2), \dots, f(B_H)\}$
4)   if $f(C_i) \le f(B_{MAX})$ do
5)      $B_{MAX} = C_i$;
6)      $f(B_{MAX}) = f(C_i)$;
7)   end if
8) end for
9) Update $P(t)$ by Eq.(10) and Eq.(11)
```

Fig. 2. The new probability update scheme

3.1 The New Probability Vector Update Scheme

Contrary to the traditional PBIL [9] that updates $\boldsymbol{P}(t)$ by shifting it towards the best individual $\boldsymbol{B}(t)$, our PBIL concerns a set of best so far individuals $S_{BSF} = \{\boldsymbol{B}_1,\dots, \boldsymbol{B}_H\}$, where $H \ge 1$ is a constant number. Initially, $\boldsymbol{P}(t)$ is sampled H times to create H individuals to form S_{BSF}. At each generation, when a number of fitter individuals appear, we update S_{BSF} by replacing those with worse fitness values in S_{BSF} by the fitter ones. Then, the statistical information of S_{BSF}, i.e. $\boldsymbol{P}_{BSF}$, is extracted and used to update $\boldsymbol{P}(t)$, as shown in formula (10) and (11).

$$\boldsymbol{P}_{BSF} = \frac{1}{\mathrm{H}} \cdot \sum_{k=1}^{\mathrm{H}} \boldsymbol{B}_k \tag{10}$$

$$\boldsymbol{P}(t) = (1.0 - \alpha) \cdot \boldsymbol{P}(t) + \alpha \cdot \boldsymbol{P}_{BSF} \tag{11}$$

Given an individual X, we denote its corresponding fitness value by $f(X)$. The procedure of the new probability vector update scheme at generation t is shown in Fig.2. Note that this new update scheme generalizes the update scheme in standard PBIL. When H = 1, it is equivalent to a standard PBIL where only one best so far individual, i.e. $\boldsymbol{B}_1$, is maintained in S_{BSF}.

3.2 The Use of All-One Vector

As the problem concerned is highly constrained, $\boldsymbol{P}(t)$ in the initialization of PBIL may not be able to create feasible individuals, and thus deteriorates the effectiveness and efficiency of PBIL. Kim *et al* [4,5] significantly improved the performance of their GA by inserting an all-one vector into initial population to enable that all merging nodes are active, and their GA begins with at least one feasible solution.

Inspired by the above idea, we employ all-one vector(s) in the probability vector update scheme to improve the performance of the proposed PBIL. The all-one vector compensates for the absence of feasible individuals in S_{BSF} in the initialization of our algorithm. For example, if there are u ($0 < u \le$ H) infeasible individuals in S_{BSF}, these individuals are replaced by u all-one vectors. Note that, different from the problem concerned in [4,5], our problem also considers the delay constraint. Therefore, all-one vector may still be infeasible as the delay constraint may not be met.

3.3 The Structure of the Proposed PBIL

We use the graph decomposition method proposed in [4,5] to transform the given network G into a secondary graph G_D. In PBIL, each individual X corresponds to a secondary graph G_D. Each bit of X is associated with one of the newly introduced links between the so-called auxiliary nodes in G_D. Bit '1' and '0' means its corresponding link exists and does not exist in the secondary graph G_D, respectively.

Fitness evaluation measures each obtained G_D. For an individual X, we first check if a feasible NCM subgraph $G_{NCM}(s,T)$ can be found from its corresponding G_D. For each sink $t_k \in T$, we use the Goldberg algorithm[11], a classical max-flow algorithm, to compute the max-flow between the source s and t_k in the corresponding G_D. If all d max-flows are at least R, for each sink t_k we select R least-delay paths from all link-disjoint paths obtained from s to t_k. All the selected paths are mapped to G_D to form the $G_{NCM}(s, T)$. If the maximal path delay in the $G_{NCM}(s,T)$ satisfies the delay constraint, i.e. $D(G_{NCM}(s,T)) \leq \Omega$, we set the number of coding links in $G_{NCM}(s,T)$ to $f(X)$. If $G_{NCM}(s,T)$ cannot be found or it violates the delay constraint, X is infeasible and we set a very large fitness value Ψ to $f(X)$ (in this paper, $\Psi = 50$).

The procedure of the proposed PBIL is shown in Fig.3. The termination criteria are subject to two conditions: 1) a coding-free subgraph is obtained, or 2) the algorithm reaches a pre-defined number of generations.

```
1)  Initialization
2)      Set $t = 0$;
3)      For $i = 1, 2, \dots, L$, set $P_i^t = 0.5$
4)      Generate a sampling set $S(t)$ of $N$ individuals from $\boldsymbol{P}(t)$
5)      Generate a set $S_{BSF}$ of H individuals by sampling $\boldsymbol{P}(t)$
6)      Replace infeasible individuals in $S_{BSF}$ by all-one vectors
7)  repeat
8)      Set $t = t + 1$
9)      Evaluate the samples in $S(t-1)$
10)     Update $\boldsymbol{P}(t)$ by using the probability vector update scheme (Fig.2)
11)     Mutate $\boldsymbol{P}(t)$ by Eq.(9)
12)     Generate a set $S(t)$ of $N$ samples by $\boldsymbol{P}(t)$
13) until termination condition is met
```

Fig. 3. Procedure of the proposed PBIL

4 Numerical Experiments and Discussions

We evaluate the performance of the following two algorithms:

- GA: the simple GA with binary link state encoding, tournament selection, uniform crossover, simple mutation and all-one vector inserted into initial population [5].
- PBIL-H(*num*): the proposed PBIL with the new probability vector update scheme and the use of all-one vector(s), where *num* is the integer set to H.

Simulations have been carried out upon two networks, i.e. a 15-copies network (with $R = 2$) adopted in [5] and a random directed network (with 60 nodes, 150 links, 11 sinks, and $R = 5$). The propagation delay over each link $D(e)$ is uniformly distributed from 2*ms* to 10*ms* and the time consumed for coding Δ_c is set to 2*ms*. For each topology, we set two different delay constraints, a severe one and a loose one, i.e. for the 15-copies topology, 136*ms* and 300*ms*, respectively; for the 60-nodes topology 71*ms* and 300*ms*, respectively. The solution lengths in 15-copies and 60-nodes are 176 bits and 235 bits, respectively. In all scenarios, the population size and the pre-defined termination generation is set to 40 and 300, respectively. In GA, tournament size, crossover and mutation probabilities are set to 2, 0.8 and 0.01, respectively. In PBIL-H(*num*), $\alpha = 0.1$, $pm = 0.02$, and $\sigma = 0.05$. To study how H affects the performance of our PBIL, we set 1, 8, 16, 24, 32, and 40 to *num*, respectively. All experimental results were collected by running each algorithm 20 times. The performance analysis is based on the following criteria:

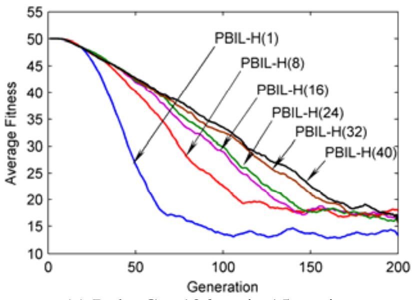
- The evolution of the average fitness. Note that the termination condition here is that algorithm stops after 200 generations.
- The successful ratio (SR) of finding a coding-free subgraph (i.e. subgraph without coding performed). Note that the optimal solutions in our experiments are solutions that produce coding-free subgraphs.
- The average best fitness (ABF). It is the average value of the obtained best fitness values in 20 runs.
- The average computational time (ACT) to run an algorithm.

Table 1 shows the results of GA and PBIL-H(1) on the 15-copies and 60-nodes networks. GA is beaten by PBIL-H(1) in every case, showing to be ineffective in solving the problems concerned. In terms of SR, the maximum value GA obtains is 60% while PBIL-H(1) has at least 80%. In particular, for 15-copies network, GA even cannot reach coding-free subgraph(s). In contrast, the performance of PBIL-H(1) is stabilized and with high-quality, which shows PBIL-H(1) is more effective compared with GA. We also see that PBIL-H(1) performs better than GA with respect to ABF and sometimes the superiority is substantial, e.g. in severely constrained cases.

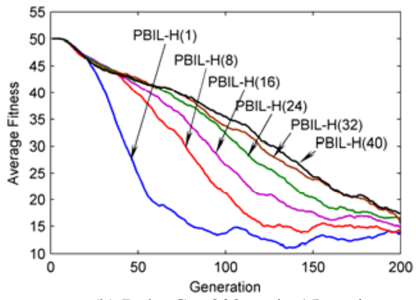
Fig.4 compares the average fitness obtained by PBIL-H(*num*) for the 15-copies and 60-nodes networks. We find that the convergence characteristics of these algorithms are affected by H, i.e. the number of best so far individuals kept in S_{BSF}, in such a way that the larger H, the slower convergence. As aforementioned, the statistical information of S_{BSF} is used to update the probability vector. With more individuals in S_{BSF}, the contribution to adjust the probability vector from a single individual in S_{BSF} becomes less. Therefore, a larger H is more likely to slow down the convergence of the proposed algorithm. However, PBILs with a smaller H may suffer from rapid diversity loss and converge to local optima.

Table 1. Results by GA and PBIL-H(1)

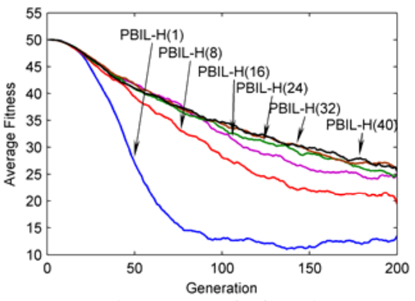
Evaluation Criteria		15-copies network		60-nodes network	
		$\Omega = 136$ *ms*	$\Omega = 300$ *ms*	$\Omega = 71$ *ms*	$\Omega = 300$ *ms*
SR (%)	GA	0.0	0.0	5.0	60.0
	PBIL-H(1)	100.0	95.0	80.0	80.0
ABF	GA	50.0	9.05	47.5	1.00
	PBIL-H(1)	0.0	0.05	0.20	0.20



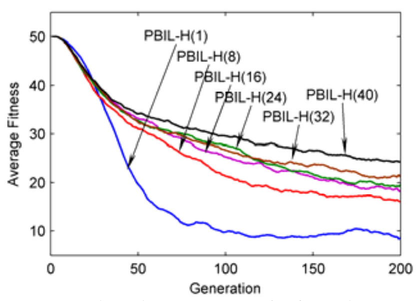
(a) DelayC = 136 *ms* in 15-copies



(b) DelayC = 300 *ms* in 15-copies



(c) DelayC = 71 *ms* in 60-nodes

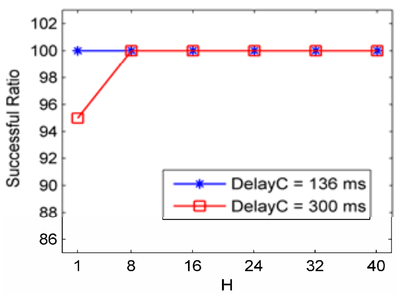


(d) DelayC = 300 *ms* in 60-nodes

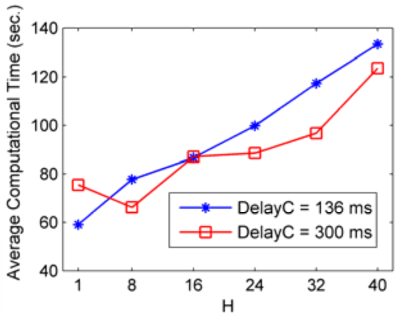
Fig. 4. Average fitness vs. generation. DelayC denotes the delay bound.

On the same network with different delay bounds, the performances of PBIL-H(*num*) deteriorate on severely constrained cases. For example, PBIL-H(1) for the 60-nodes network with smaller delay bound in Fig.4(c) obtains a worse average fitness than that of the same network with larger delay bound in Fig.4(d). Obviously, this is due to that there are less feasible solutions in the solution space of the severely constrained topologies compared to that of loosely constrained cases.

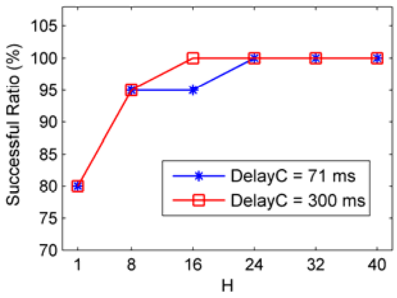
Fig.5 shows the effect of H to SR and ACT in the 15-copies and 60-nodes networks. In Fig.5(a) and (b), we notice that the larger the value of H, the higher the SR, and the larger ACT. In the case of severe delay constraint, the ACT of PBIL-H(*num*), *num* = 1, 8, 16, 24, 32 and 40, become increasingly worse. This phenomenon demonstrates the tradeoff between diversification and intensification. The more the best so far individuals are maintained in S_{BSF}, the better the diversity is kept, thus avoiding pre-maturity. However, larger H, on the other hand, slows down the convergence, where PBIL cannot efficiently exploit and find an optimal solution from a certain area of the solution space yet consumes a large amount of computational time. Concerning SR in Fig.5(c), we also notice that the larger the value of H, the higher the SR in cases with different delay constraints. On the other hand, in Fig.5(d), the ACT falls first and rises up then. This is because with H increasing, the global exploration ability of the algorithm is gradually improved, leading to the ACT to obtain optimal solutions being reduced. However, during the process, the convergence of PBIL becomes flattened, inherently prolonging the ACT. At the beginning, the reduction of ACT caused by high global exploration is more than the addition of ACT caused by slow convergence, so the actual ACT decreases. Once the reduction of ACT is less than its addition, the



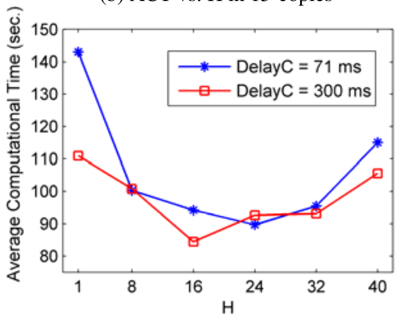
(a) SR vs. H in 15-copies



(b) ACT vs. H in 15-copies



(c) SR vs. H in 60-nodes



(d) ACT vs. H in 60-nodes

Fig. 5. The effect of H

actual ACT undoubtedly grows up. From above analysis, we find that if H is properly selected, e.g. set H = 8 in 15-copies network and H = 24 in 60-nodes network, rapid convergence and global search capability can be achieved simultaneously.

5 Conclusions

In this paper, we investigate for the first time the delay constrained minimization problem on network coding operations. A population based incremental learning algorithm with a new probability vector update scheme and all-one vector(s) is proposed. The new update scheme makes use of a group of best so far individuals to adjust the probability vector. The number of best individuals needs to be carefully devised so that high performance and low computational time are achieved at the same time. Besides, all-one vectors are adopted to drive the probability vector towards feasible solutions at early evolutionary generations, which greatly improve the performance of our proposed algorithm. Simulation results demonstrate that the proposed algorithm is highly effective in solving the problem concerned.

Acknowledgements

This research is supported by China Scholarship Council and The University of Nottingham, UK.

References

1. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung, R.W.: Network coding flow. IEEE Trans. Inform. Theory 46(4), 1204–1216 (2000)
2. Li, S.Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Trans. Inform. Theory 49(2), 371–381 (2003)
3. Kim, M., Ahn, C.W., Médard, M., Effros, M.: On minimizing network coding resources: an evolutionary approach. In: Proc. NetCod. (2006)
4. Kim, M., Médard, M., Aggarwal, V., O'Reilly, V., Kim, W., Ahn, C.W., Effros, M.: Evolutionary approaches to minimizing network coding resources. In: Proc. Inforcom (2007)
5. Kim, M., Aggarwal, V., O'Reilly, V., Médard, M., Kim, W.: Genetic representations for evolutionary minimization of network coding resources. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 21–31. Springer, Heidelberg (2007)
6. Fragouli, C., Soljanin, E.: Information flow decomposition for network coding. IEEE Trans. Inform. Theory 52(3), 829–848 (2006)
7. Langberg, M., Sprintson, A., Bruck, J.: The encoding complexity of network coding. IEEE Trans. Inform. Theory 52(6), 2386–2397 (2006)
8. Bhattad, K., Ratnakar, N., Koetter, R., Narayanan, K.R.: Minimal network coding for multicast. In: Proc. ISIT 2005 (2005)
9. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University (1994)
10. Larranaga, P., Lozano, J.A.: Estimation of distribution algorithms: a new tool for evolutionary computation. Kluwer, Norwell (2002)
11. Goldberg, V.: A new max-flow algorithm. Technical Report MIT/LCS/TM-291, MIT (1985)

Extremal Optimization Applied to Task Scheduling of Distributed Java Programs

Eryk Laskowski[1], Marek Tudruj[1,3], Ivanoe De Falco[2], Umberto Scafuri[2], Ernesto Tarantino[2], and Richard Olejnik[4]

[1] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[2] Institute of High Performance Computing and Networking, ICAR-CNR, Naples, Italy
[3] Polish-Japanese Institute of Information Technology, Warsaw, Poland
[4] Computer Science Laboratory, University of Science and Technology of Lille, France
{laskowsk,tudruj}@ipipan.waw.pl, richard.olejnik@lifl.fr,
{ivanoe.defalco,umberto.scafuri,ernesto.tarantino}@na.icar.cnr.it

Abstract. The paper presents new Java programs scheduling algorithms for execution on clusters of Java Virtual Machines (JVMs), which involve extremal optimization (EO) combined with task clustering. Two new scheduling algorithms are presented and compared. The first employs task clustering to reduce an initial program graph and then applies extremal optimization to schedule the reduced program graph to system resources. The second algorithm applies task clustering only to find an initial solution which is next improved by the EO algorithm working on the initial program graph. Both algorithms are also compared to an EO algorithm which does not use the clustering approach.

Keywords: distributed systems, scheduling, evolutionary algorithms.

1 Introduction

Extremal Optimization (EO) developed in 1999 by Boettcher and Percus [2] offers an evolutionary optimization method, which features fast convergence and very small memory requirements. EO works based on developing a single solution composed of a number of components, each of which is a variable of the problem. In this respect, EO is strongly competitive to genetic algorithms. Two separate fitness functions are used to evaluate the components and the global solution quality. In EO, the worst component is randomly updated, so that the solution is transformed into a new acceptable solution. A probabilistic version of EO has been designed [3] which aims in avoiding the local minimum phenomenon.

Optimization of the execution time of Java programs has already attracted researchers' attention [7]. The optimization is done at program runtime with the use of centralized or distributed load monitoring. Java program task scheduling on Grids is also reported in [4]. It requires adequate computational and communication load metrics proposed in [12]. Optimal execution of Java programs should take into account an optimized initial schedule of Java distributed application. This problem has not been sufficiently covered in current literature,

© Springer-Verlag Berlin Heidelberg 2011

although some papers propose an initial optimization including Java objects static clustering [9] or distributed byte-code clustering [10] in a set of JVMs.

This paper follows our earlier research reported in [5] where we have studied the use of the probabilistic EO for initial mapping of Java programs to distributed systems of multi-core processors. Our algorithms have confirmed that the quality of schedules provided by EO is similar to that of the algorithms based on the Earliest Task First heuristics (ETF). In this paper, we present improved versions of the initial Java program placement algorithm, which uses the probabilistic EO approach. The original feature of our algorithms is that we preceded the EO essential optimization phase by an initial clustering of the program macro data flow graphs. Two approaches to the use of task clustering in EO have been studied. With the first approach, task clustering is performed to find a reduced version of an initial program macro data flow graph, which is next processed by the EO algorithm to provide the final program schedule. With the second approach, the clustering algorithm is used to find only the initial solution for the EO, i.e. a mapping of the program tasks to processor nodes; however, the program graph considered by the EO is the initial macro data-flow graph of the program. The two approaches are compared by experiments based on graph execution simulator. The experiments have shown that the second approach gives a better quality results. Environment monitoring (system observation) predicts CPU and network services availability based on current CPU load and network utilization which are used in the proposed algorithms.

In our experiments, we have used the ProActive Java framework for cluster and Grid computing [1] as distributed programs execution management support. It provides a distributed Java API and a set of tools for program management in different environments such as desktop, SMP, LAN, clusters and Grid.

The rest of the paper is composed of 4 parts. The first part presents the program representation and the executive system features. Next, the extremal optimization principles are outlined. Then, the proposed and discussed versions of the extremal optimization for Java program scheduling are described. Finally, experiment results with programs optimized for execution on cluster of JVMs based on multicore processors are presented.

2 Program Representation and System Environment

In the paper, we are interested in initial deployment optimization of distributed Java programs, which can be represented as directed acyclic graphs (DAGs). An application program is described by a weighted directed acyclic graph $G_{dag} = \{P, E\}$, where P is a set of **communicating task** nodes, and E is a set of **data transfer** edges between tasks. Each task $p_k, k \in \{1 \ldots |P|\}$ has a weight γ_k which represents the number of instructions to be executed. These weights are determined either during the sample execution of the program for some representative data or are provided by application developer. An edge weight

ψ_{km} represents the amount of data to be sent from task k to another m-th task. Similarly, these weights can be sampled or provided explicitly by a developer. A program is executed according to the *macro data-flow* model. Tasks start their execution when all needed data arrived through task incoming edges. When task is finished, it sends produced data to the succeeding tasks. The graphs are static and deterministic. We assume that, to preserve the DAG constraints, all loops are unrolled or encircled inside the body of a single task.

The target executive system consists of N computing resources (nodes). Each node, identified by an integer value in the range $[0, N-1]$, is a multicore processor. We assume that all cores in a single node i, whose number is denoted as κ_i, are homogeneous. The current status of system resources is given by the **node power** α_i which is the number of instructions computed per time unit in a core of the node i and average load of each core $l_i^k(\Delta t)$ in a particular time span Δt: $l_i^k(\Delta t)$ ranges in $[0.0, 1.0]$, where 0.0 means a core with no load and 1.0 a core loaded at 100%. Thus $(1 - l_i^k(\Delta t))\alpha_i$ represents the power of the core k of the node i available for the execution of the tasks scheduled by our algorithm. The **communication bandwidth** between any pair of nodes i and j is denoted as β_{ij}. The current status of the system is supposed to be contained in tables based either on statistical estimations in a particular time span or gathered by tracking periodically and by forecasting dynamically resource conditions.

ProActive framework. In the presented work, execution of distributed Java programs is done employing the ProActive framework for cluster and Grid computing [1]. ProActive is a Java middleware library (available under GPL open source license) providing an API for parallel, distributed, and multi-threaded computing, also with support for mobility and security. It is based on the Active Objects design pattern and allows for simplified and uniform programming of Java applications distributed on Local Area Networks (LANs), Clusters, Internet Grids and Peer-to-Peer Intranets.

A distributed ProActive application is composed of a set of active objects. An Active Object is implemented as a standard Java object with an attached thread of control. Incoming method calls are stored in a queue of pending requests in each active object, which decides in which order to serve them. Thus, method calls sent to active objects are asynchronous with transparent future objects and the synchronization handled by a *wait-by-necessity* mechanism. The communication semantics depends upon the signature of the method, with three possible cases: synchronous invocation, one-way asynchronous invocation, and asynchronous invocation with future result.

The program DAG representation corresponds to ProActive distributed application in which a task node is a thread of an Active Object and an edge is a method call in another Active Object. The execution of an Active Object method, which constitutes the task, can start when this object collected all necessary data. During execution, an Active Object communicates only with local objects. At the end of execution, the method sends data to the node successors.

3 Extremal Optimization Algorithm Principles

Extremal Optimization is an important nature inspired optimization method. It was proposed by Boettcher and Percus [2] following the Bak–Sneppen approach of self organized criticality dynamic [11]. It represents a method for NP–hard combinatorial and physical optimization problems [3, 2]. It is also 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 $Neigh(S)$.

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 $S' \in Neigh(S)$. 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: $\phi_\pi(1) \leq \phi_\pi(2) \leq \ldots \phi_\pi(n)$, 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: $p_k / k^{-\tau}$, $1 \leq k \leq n$ for a given value of the parameter τ. Finally, at each update a generic

Algorithm 1. General EO algorithm

```
initialize configuration $S$ at will
$S_{\text{best}} \leftarrow S$
**while** maximum number of iterations $\mathcal{N}_{\text{iter}}$ 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 $S' \in Neigh(S)$ such that $s_j$ must change
    accept $S \leftarrow S'$ unconditionally
    **if** $\Phi(S) < \Phi(S_{\text{best}})$ **then**
        $S_{\text{best}} \leftarrow S$
    **end if**
**end while**
**return** $S_{\text{best}}$ and $\Phi(S_{\text{best}})$
```

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 $S' \in \mathit{Neigh}(S)$ unconditionally. The only parameters are the maximum number of iterations $\mathcal{N}_{\mathrm{iter}}$ and the probabilistic selection value τ. For minimization problems τ–EO proceeds as in the Algorithm 1.

4 Extremal Optimization Applied to Java Program Optimization

Improved versions of the initial Java program placement algorithm, which uses the probabilistic EO approach are the main goals of this paper. The target problem is defined as follows: assign each task $p_k, k \in \{1 \dots |P|\}$ of the program to a computational node $i, i \in [0, N-1]$ in such a way that the total program execution time is minimized, assuming the program and system representation as described in section 2. Since tasks address non-dedicated resources, their own local computational and communication loads must be considered to evaluate the computation time of the tasks of the program to be scheduled. There exist several prediction methods to face the challenge of non–dedicated resources.

The original feature of new algorithms is that we introduced an additional clustering step before the essential, EO-based optimization phase. Two approaches to the use of the task clustering before the EO phase have been investigated. In the first approach, task clustering is performed to find a reduced version of an initial program macro data flow graph, which is next processed by the EO algorithm to provide the final program schedule. This approach aims at reduction of the overall execution time of the EO phase through reducing the size of the program graph. In the second approach, the clustering algorithm is used to find only the initial solution for the EO, i.e. a mapping of the program tasks to processor nodes; however, the graph considered by the EO is the initial macro data-flow graph of the program. The second approach aims at improving the quality of the final schedule of the program.

Using the aforementioned two approaches to an initial program graph clustering step, we propose the following extremal optimization algorithm variants:

EO-clust – an EO algorithm using the clustering to find a reduced version of the program graph, which will be next considered as an input to the EO algorithm,

EO-ini-cl – an EO algorithm using the clustering to find only a better initial solution for the EO while it starts with considering a full introductory program graph.

Both algorithms are based on the same EO heuristics, as shown in Algorithm 1. In the clustering algorithm, we have applied the DSC heuristics (see [13] for details), which is proven to be both efficient and fast.

A scheduling 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 $\mu_i = j$ means that

Algorithm 2. Program graph execution simulation

```
Mark entry task of the graph as ready at the time 0
for each core $c$ of all processors: $Availability\_time(c) \leftarrow 0$
while not all tasks are visited do
    $t \leftarrow$ the ready task with the earliest starting time
    $n \leftarrow \mu_t$ {the node of task $t$}
    $c \leftarrow$ the core of $n$ which has the earliest $Availability\_time$
    Place task $t$ on core $c$ of node $n$
    $Starting\_time(t) \leftarrow \max(Availability\_time(c), Ready\_time(t))$
    $TaskCompletion\_time(t) \leftarrow Starting\_time(t) + Execution\_time(t)$
    $Availability\_time(c) \leftarrow TaskCompletion\_time(t)$
    Mark $t$ as visited
    for all succesor task $s_i^t$ of task $t$ do
        $DRT \leftarrow TaskCompletion\_time(t) + Communication\_time(t, s_i^t)$
        if $DRT > Ready\_time(s_i^t)$ then
            $Ready\_time(s_i^t) \leftarrow DRT$
        end if
        if $TaskCompletion\_time(t) > LastParent\_time(s_i^t)$ then
            $LastParent\_time(s_i^t) \leftarrow TaskCompletion\_time(t)$
        end if
        if all data of $s_i^t$ arrived then
            mark $s_i^t$ as ready
        end if
    end for
end while
return $\max(Availability\_time)$
```

the solution S under consideration maps the i–th task p_i of the application onto processor node j. The number of processor cores is not represented inside the solution encoding, however, it is taken into account when estimating the global and local fitness functions while solving the scheduling problem. This will be explained below.

The **global fitness** in the applied EO examines the time of execution of a scheduled program. The execution time of the scheduled program is provided by a program graph execution simulator (Algorithm 2). The simulator assigns time annotations to program graph nodes based on the processor computing power availability and communication link throughput available for a given program execution. Algorithm 2 determines also the $TaskCompletion\_time(t)$ and $LastParent\_time(t)$ values for each task, which are then used during computing the local fitness function.

In the applied EO the **local fitness** function (LFF) of a task is the complete delay of task execution comparing the execution under "optimal" conditions, i.e. when there is no communication overhead nor resource contention between tasks and the task is executed on the fastest processor. We call this delay the total execution delay.

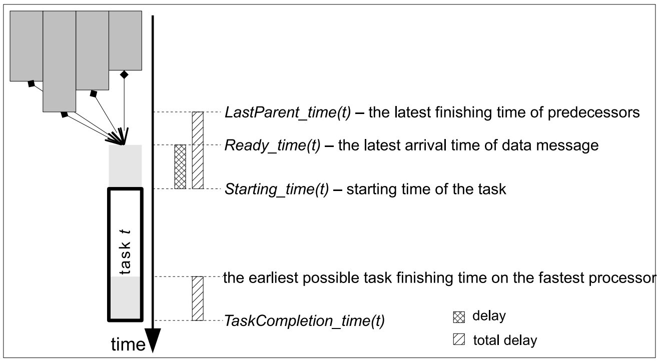

LastParent_time(t) – the latest finishing time of predecessors
Ready_time(t) – the latest arrival time of data message
Starting_time(t) – starting time of the task
task t
the earliest possible task finishing time on the fastest processor
TaskCompletion_time(t)
▨ delay
▨ total delay
time

Fig. 1. Computation of *delay* and *total delay* values for given task t

$$LFF(t) = TaskCompletion\_time(t) - LastParent\_time(t) - FastestExecution(t)$$

where $FastestExecution(t)$ is the execution time of task t on the fastest processor.

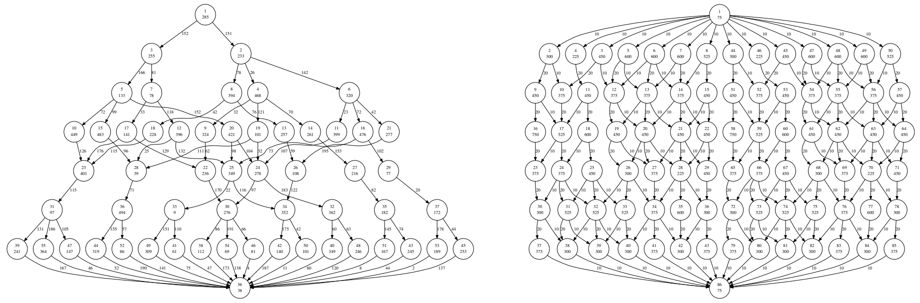
All parameters necessary for computing the value of the local fitness function are obtained during the execution of program graph simulation procedure (see Algorithm 2 and Fig. 1).

5 Experimental Results

During the evaluation of proposed extremal optimization algorithms we measured the actual execution times of application programs in a computational cluster. The application programs, represented as graphs, have been optimized before execution using an investigated algorithm.

To obtain the comparative results, we have used an additional EO algorithm which does not involve the clustering phase (**EO**) and a list scheduling algorithm with the Earliest Task First (**ETF**) heuristics. The ETF implementation is based on the description from [6]. We executed scheduled synthetic programs on a cluster of 7 homogeneous dual core processor nodes for program graph scheduling and program execution under ProActive. The values of α_i, β_i, κ_i and $l_i^k(\Delta t)$ for the cluster are measured using a benchmark tool, which is the part of our runtime framework. After an EO algorithm tuning, $\mathcal{N}_{\text{iter}}$ parameter has been set to 5000, and τ to 3.0.

During our experiments we have examined two sets of synthetic graphs and the graph of a medical application – ART algorithm (reconstruction of tomographic scans [8]). The first set of synthetic graphs consists of seven randomly generated graphs (*gen-1...3, gen-3a...d*), with layered structure, Fig. 2(a). Each task (node) of this graph represents a mixed float- and integer-based generic computation (random generation of matrix of doubles, then floating-point matrix-by-vector multiplication, then rounding to integers and integer sort) with execution time defined by node weight (the weight controls the number of iterations of the



(a) gen-3b (b) gen-m2

Fig. 2. The structure of a synthetic exemplary application graph

generic computation). The second set of synthetic graphs consists of two hand-made graphs with known optimal mappings (*gen-m1, gen-m2*), with a general structure similar to the structure of randomly generated graphs, Fig. 2(b).

Comparison of real execution times of an exemplary application, scheduled by different methods is presented in Fig. 3 and Fig. 4. The different variants of EO method obtained similar quality of initial mappings of applications. For synthetic graphs, the best results are obtained by **EO-ini-cl** algorithm, however the **EO** method is only marginally worse. For ART application graph, the **EO-ini-cl** method is the best among different EO variants. The typical execution time difference, comparing the EO and ETF algorithms, is below 10% (the only exception is *gen-m1* graph, for which only **EO-ini-cl** and **ETF** were able to find the optimal solution). The experimental results show that EO technique is able, in general, to achieve the same quality of results as classical scheduling and mapping approaches like ETF algorithms. It is a good result, taking into account the simplicity of the basic principle of extremal optimization.

In another experiment we empirically extrapolated the actual time complexity of presented algorithms. For this purpose we used a set of large, randomly generated graphs (the number of nodes from 350 to 7000), which were scheduled by extremal optimization and ETF algorithms. The actual running times confirmed the theoretical complexity of EO and ETF methods, which is approximately $C(n)$ for EO and $C(n^2)$ for ETF (where n is the size of the graph). Although time complexity of EO methods is lower than that of ETF, the actual running times of different kinds of the EO algorithms for small graphs were much longer than the running times of ETF algorithm. It could be considered as the main drawback of the EO method. Among investigated EO variants, all have similar execution time, since the additional clustering step introduces only small run-time overhead due to the dominant sequence tracking. The clustering phase applied to the **EO-ini-cl** algorithm has improved problem solution quality. For the **EO-clust** algorithm, it has reduced the EO problem solving time from 1.23 to 2.9 times (the average reduction factor for all investigated graphs was 2.27).

Experimental results indicate that extremal optimization technique can be useful for large mapping and scheduling problems when we will pay special

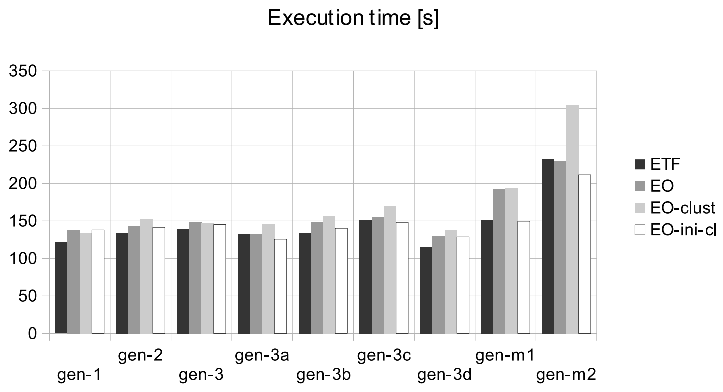


Fig. 3. The real execution times of the scheduled synthetic program graphs for different scheduling algorithms

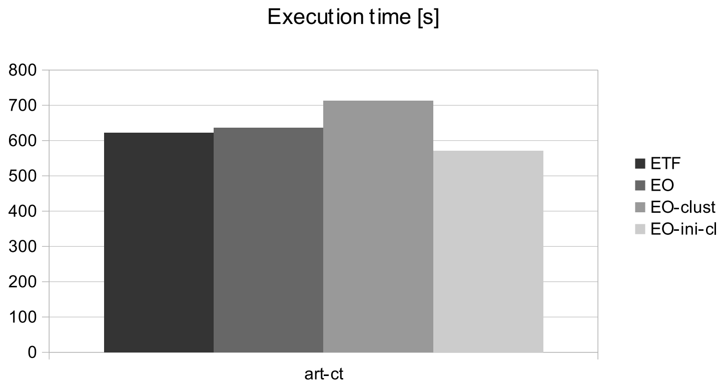


Fig. 4. The real execution times of ART program graphs for different scheduling algorithms

attention to run-time optimization of EO algorithm. For small sizes of application graphs, it is enough to use classic scheduling methods, as ETF list scheduling.

At the time of writing of this paper, we were still continuing experimental research using the presented algorithms. During these experiments, we expect to enlarge both the size and diversity of the test graph set. Future works are aimed at the optimization of the execution speed of EO-based heuristics.

6 Conclusions

The paper has presented new Java programs scheduling algorithms for execution on clusters of Java Virtual Machines (JVMs). They combine extremal optimization with task clustering. Two new scheduling algorithms were compared. The algorithm which has applied task clustering only to find an initial solution which was next improved by the EO algorithm working on the initial program graph produced better schedules.

The described extremal optimization algorithms have produced schedules whose quality was comparable to those of the ETF algorithms. The execution times of the synthetic programs corresponding to the scheduled graphs were close to the execution times in a real system.

References

1. Baude, et al.: Programming, Composing, Deploying for the Grid. In: Cunha, J.C., Rana, O.F. (eds.) GRID COMPUTING: Software Environments and Tools. Springer, Heidelberg (2006)
2. Boettcher, S., Percus, A.G.: Extremal optimization: methods derived from coevolution. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 825–832. Morgan Kaufmann, San Francisco (1999)
3. Boettcher, S., Percus, A.G.: Extremal optimization: an evolutionary local–search algorithm. In: Bhargava, H.M., Kluver, N.Y. (eds.) Computational Modeling and Problem Solving in the Networked World, Boston (2003)
4. Don, F.: A taxonomy of task scheduling algorithms in the Grid. Parallel Processing Letters 17(4), 439–454 (2007)
5. De Falco, I., Laskowski, E., Olejnik, R., Scafuri, U., Tarantino, E., Tudruj, M.: Extremal Optimization Approach Applied to Initial Mapping of Distributed Java Programs. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010. LNCS, vol. 6271, pp. 180–191. Springer, Heidelberg (2010)
6. Hwang, J.-J., et al.: Scheduling Precedence Graphs in Systems with Interprocessor Communication Times. Siam J. Comput. 18(2), 244–257 (1989)
7. Jimenez, J.B., Hood, R.: An Active Objects Load Balancing Mechanism for Intranet. In: Workshop on Sistemas Distribuidos y Paralelismo, WSDP 2003, Chile (2003)
8. Kak, A.C., Slaney, M.: Principles of Computerized Tomographic Imaging. IEEE Press, New York (1988)
9. Laskowski, E., et al.: Java Programs Optimization Based on the Most–Often–Used–Paths Approach. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 944–951. Springer, Heidelberg (2006)
10. Laskowski, E., et al.: Byte-code scheduling of Java programs with branches for Desktop Grid. Future Generation Computer Systems 23(8), 977–982 (2007)
11. Sneppen, K., et al.: Evolution as a self–organized critical phenomenon. Proc. Natl. Acad. Sci. 92, 5209–52136 (1995)
12. Toursel, B., Olejnik, R., Bouchi, A.: An object observation for a Java adaptative distributed application platform. In: International Conference on Parallel Computing in Electrical Engineering (PARELEC 2002), pp. 171–176 (September 2002)
13. Yang, T., Gerasoulis, A.: DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors. IEEE Trans. on Parallel and Distributed Systems 5(9) (1994)

Data-Centered Scheduling for Addressing Performance Metrics on WSN

Lia Susana d.C. Silva-Lopez and Jonatan Gomez

Universidad Nacional de Colombia - Bogota. Department of Systems Engineering
ALIFE Research Group
{lcsilval,jgomezpe}@unal.edu.co

Abstract. In this paper, a novel combination of cross-layer strategies for addressing timeliness, handling latency times on a data-centred way, and improving energy management in a non-mobile WSN scenario, is proposed.In this way, a set of performance metrics (for timeliness, latencies and energy management) are introduced and used for evaluating Periodic Scheduling and Simplified Forwarding strategies. The WSN is modelled as an four states Asynchronous Cellular Automaton with irregular neighbourhoods. Therefore, only information from local neighbourhood is needed for communication between nodes. Our results show that the proposed strategies and performance metrics are useful for sensing data accurately, without excessive oversampling.

1 Introduction

A Wireless Sensor Network (WSN) is a kind of distributed system, in which nodes are provided with a CPU, a wireless transceiver and a set of sensors. WSN nodes have computing capabilities, can communicate with each other to execute a given task, and are deployed in an area, where a phenomenon takes place.

Cellular Automata (CA) are a natural way to model WSN. CA are biologically inspired models of cellular growth, proposed in the early 1950s by Stanislaw Ulam and John Von Neumann. A WSN can be seen as an automaton, in which local states change, depending on information gathered from a set of neighbours. The use of CA to model WSN allows interesting features, but some limitations as well; a node, just like a cell, has a limited number of immediate neighbours, and changes its state according to the states of its neighbours, or its own state. For example, efficient network clustering and long term global energy administration were obtained, using an irregular automaton that learns clusters in [1].

Using a synchronous cellular automaton, on real WSN scenarios, require the whole network to stop every process, in order to allow each node to perform the calculation of its next state. Such situation is not desirable, because global coordination is required, and the advantage of CA lies in the absence of global supervision. Asynchronous CA are a more realistic representation of the way transitions are performed on WSN. Li in [2], compares Synchronous and Asynchronous CA-based models of WSN.

© Springer-Verlag Berlin Heidelberg 2011

The absence of global coordination will allow WSN to perform real unattended monitoring and control. For unattended control, references of performance on each required feature, are needed. There are several ways to measure performance on WSN features, depending on certain features [3]. One way, is to use performance metrics.

Examples performance features are Energy Management, Latency Times, and Precision on delivered information (accuracy, timeliness). By far, the most popular performance feature in WSN literature is Energy Management, since either directly or indirectly, all WSN-related work has to do something with energy [4]. Latencies and robustness are also somewhat popular, especially the first one. Latencies refer to waiting times in communication. They have been explored for a long time, and usually dealt with from the point of view of the network layer [5]. Geographic information is also used for improving Latencies and Robustness [6]. Literature about protocols that address timeliness or accuracy, is not as frequent. Most approaches tend to regard timeliness and accuracy indirectly, if ever discussed [7].

In this paper, Energy Management, Latencies and Timeliness are addressed in a data-centred way, with a scheduling strategy we present (POLA), in which the WSN is modelled as an Asynchronous Cellular Automaton. To measure such features in a data-centred context, performance metrics for each feature are presented in Section 2; in Section 3, the scheduling strategy, in which WSN nodes behave as cells of a Cellular Automaton, is presented; in Section 4, experimental settings and simulation results are presented; here, POLA is tested on different datasets, and compared to Periodic Scheduling with Simplified Forwarding, by using the performance metrics proposed in Section2. Finally, in Section 5, contributions, conclusions, and future directions are stated.

2 Proposed Performance Metrics

There are many ways to evaluate performance on WSN features. Three features are evaluated in this paper: Timeliness, Latencies and Energy Management. This metrics define a framework to compare WSN strategies; such framework can be used with any existing protocol to evaluate cross-layer features.

In the context of this paper, Timeliness refers to how accurately is a variable sensed on a temporal context, or how timely measuring intervals are. On any application, data is generated at certain intervals. If too much samples are taken, then energy is wasted on oversampling, but the chance of catching a rapid and unexpected change in data generation intervals, is increased. To measure how much over- or under- sampling occurred, we propose to use the Hamming Distance between the expected number of samples, and the actual number of samples ($Hamming$ in Table 1).

The Hamming metric however, does not hold information on how well the envelope of the data was captured by the system. This is achieved by the Normalized Difference between Trapezoidal Regions (N_{trap}) metric, in Table 1. The Trapezoidal rule in equation 1, is used to estimate the area of the region under

Table 1. Expressions for Timeliness Metrics

Hamming	N_{trap}	$AST_{shifting}$
S_g - S_s	$\frac{Trap(g)-Trap(s)}{Trap(g)}$	$\frac{1}{T}\sum_{k=0}^{L}(\alpha-\delta(k))$

the envelope of each sample set; then, the absolute difference between regions is normalized to the area of the trapezoidal region of Generated samples.

Even when proper sampling is performed by considering $Hamming$ and N_{trap}, those samples may be shifted some time units, from the moment they should have been taken. Measuring how much shifting occurred is especially important when sampling intervals are not equally spaced. To measure an average of how much shifting occurred, a new metric is proposed in this paper, named Average Sampling Time Shifting ($AST_{shifting}$), in Table 1.

$$Trap(x) = \frac{1}{Interval_{Size}} \sum_{k=0}^{S_x-\alpha} |(\alpha)(D_x(k) - D_x(k+\alpha))| \tag{1}$$

S_x is the number of samples in the time interval in which the metric is going to be evaluated; it can be an expected number of samples, or a number of samples retreived by an Application Layer. $D_x(k)$ is the data sample released by Data Generator on instant k. D_x(k) is the data sample gathered by Application Layer on instant k. α is the period in which a new sample is released by the Data Generator. α is replaced by $\delta(k)$ when applying the metric, to represent the varible interval in which Application layer senses; for every instant k, there is a $\delta(k)$ value that depends on the change rate of sensed data. T is the total simulation time of the test, and L is the total number of samples emitted by a Data Generator, or Sensed by an Application Layer (the smaller is chosen).

On event-driven scenarios, which are common on WSN applications, events are generated each time a node polls for data (not always periodically). Considering this, latency measurements in event driven scenarios, make sense only if measured in close observation of sensing events. To evaluate how well an event driven strategy performs on latencies, the Hop Count Mode (HCmode) and Maximum Hop Count (MHC) are used. The HCmode is obtained by finding the highest value in the Hop Count Histogram of a run. The MHC, is the biggest Hop Count value reached by a message issued to a Sink node, from a node that senses. In the context of this paper, a **Hop** refers to the next node that receives a message (it can be a final destination, or a node in between).

As for Energy Management, the metrics are Overall Communication Energy (OCEe), and Mean and Variance for the Vector of Individual Communication Energy (ICEe). Units are Joules. OCEe, is the sum of all consumptions related to communication tasks, in the whole network. It represents a global perspective of how much energy a strategy spends, in terms of communication packets. In POLA, is the sum of the OCE of all packet types described in Section 3.

Equation 2 must be used with each packet type of the protocol. The sum is multiplied by two, because transmission uses energy in the node that sends, and the node that receives data. Here, Packet size (Pck_{size}) is the size of the packets sent by the network layer; Bit rate (Bit_{rate}) is a parameter in the physical layer of the enabling technology that allows access to the media; Clock Delays (D_{clk}), are the sum of all delays related to transmission and reception of information; Instantaneous Power Consumption (I_p) is the amount of power used by communications, in a time interval.

$$\frac{OCEp}{2} = \sum_{k=0}^{Packets-1} \left(\frac{Pck_{Size}}{Bit_{rate}} \frac{[bits]}{[\frac{bits}{s}]} + D_{Clk}[s]\right) * I_p[\frac{J}{s}] \tag{2}$$

As for the mean and variance of the vector of ICEe, each element in the vector, is the sum of all energetic consumptions associated to communication tasks in an individual node, (sending and receiving data). Each element in the vector can be calculated by summing consumptions of sent and received packets of each node, replacing $Packets - 1$ in the right side of Equation 2 with $sentPckts - 1$ and $receivedPckts - 1$.

3 Data-Centred Scheduling

The Data-Centred Scheduling strategy presented in this paper, is named POLA, and is designed to work in a non-mobile environment. The similarity between CA and WSNs, was used to design the POLA scheduling strategy as a state machine for every node. Nodes are placed as cells in the grid of an Asynchronous CA, with irregular neighbourhoods.

Each cell is composed by two layers that interact through messages: Application Layer (APP) and Network Layer (NWK). Each cell in the CA has four states: initialization, data sensing, data forwarding, and neighbourhood updates. The Initialization state, allows POLA to fetch input parameters, most of them related to adaptation rates, and features of the data, and buffer size for slope calculation: APP fetches its input parameters which are: Higher and Lower thresholds (Thr_h and Thr_l), Data buffer size (*buff*), Step size (*Step*) and Initial sensing rate (I_{Srate}). Then, APP sends an initialization message to NWK. An Initial sensing rate (I_{Srate}), defines when will be sensed the first sample. Sensing rates may be adjusted each time data is sensed, depending on data slopes. Then, NWK sends a NWK_{init} message into the wireless medium, and waits for some seconds. During that time, the NWK is expecting to receive acknowledgements (ACK) of the neighbours that listened to the message, or to receive NWK_{init} messages. If NWK_{init} or acknowledgement messages are received, NWK adds the senders as neighbours, and responds to NWK_{init} messages with acknowledgements. If the neighbourhood is full, a "full" message is sent in response, in the type identifier field of an ACK, so that the other nodes get the chance to remove the "full" node, since it can not accept any more neighbours.

Data Sensing states happen internally and independently of other nodes, affected exclusively by data change rates, in the APP. A data unit is sensed and stored into a buffer of size *buff*, that behaves as a FIFO queue. The next sensing interval is obtained by comparing the slope of the buffer, which is the change rate of the sensed data at that time, with the higher threshold (Thr_h) and the lowest threshold (Thr_l). If the value of the slope is between those two thresholds, then sensing rates continue as they where; if the value of the slope is higher than (Thr_h), it means that there has been a rapid increase of the measured variable's values and more frequent observations better follow those changes, therefore the sensing rate is increased in *Step* units, which means the next sensing period will be the result of substracting *Step* from the actual sensing period.

After data is sensed, it must be directed to a Sink node. That is the task of the Data Forwarding state. Some nodes are expected to have a Sink node in their neighbourhood, otherwise data could not be forwarded to Sink nodes. A node knows if it or its immediate neighbours have access to a Sink node, because it stores a vector of Sink Flags, which contain a true or false value, when a neighbour has or does not has direct access to a Sink node. Even if the POLA strategy is designed for a static scenario, there may be situations where one or several nodes are unreachable for a long time, like the case where a node runs out of energy, or removed for some reason.

Checking the neighbourhood for integrity, prevents messages from not being delivered when the next expected hop, is not available. A Neighbourhood Update state, checks for changes in the neighbourhood of a node, at randomly defined intervals between two application-defined values (*minUpd*-*maxUpd*), and updates the neighbour list. Updating the neighbourhood implicates a communication process with the new neighbours. If a node no longer responds with an Acknowledgement (ACK) to the Neighbourhood Update message (rU) and a new node appears, the new node takes the place of the old node on the list.

POLA has four types of packages: Datapacket (has a sink flag, data, own address and the address of the other node), Ack/NeighUpdate packet (has a type identifier, a sinkflag, own address and the address of the next node), and NWKinit packet (has a type identifier, and the next time in which an application message will be sent). Address Fields are 12 bits long. Nodes in the WSN are the cells of the grid in the CA. Sink nodes only receive information, and return acknowledge messages, when initialization or neighbourhood updates are performed. States change depending on internal circumstances of each node and its neighbourhood.

4 Experiments and Results

In this section, the performance of POLA and Simplified Forwarding (SF) NWK with Periodic Sensing Scheduling (PSS) APP, are compared using the performance metrics presented in this paper, on different input datasets. In SF, data is sent to the neighbour that first acknowledges a ping in order to forward it to a Sink node.

Three different datasets are used for such comparison; two of them are gathered from physical measurements, and one of them is artificially generated using a Script. The first two datasets, are real measurements of Temperature and Wind

Table 2. Features of the three Datasets: Number of samples, Range of Data, Mean and Standard Deviation, Max and Average Slope. Minimum slopes are zero on all cases.

Dataset	Samples	Range	Mean	StDev	MxSlope	AvgSlope
Temperature	517	2.2-33.3	18.9	5.81	20.9	4.66
Wind Speeds	517	0.4-9.4	4.02	1.79	8.1	1.76
Artificial	304	0.1-106.8	35.2	31.05	3.2	1.40

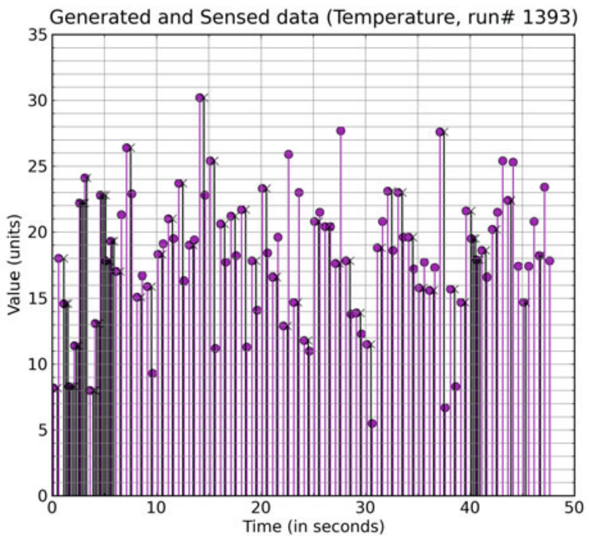
speeds, as described in [8]. Data was gathered in the Montesinho natural park in Portugal, between January 2000 to December 2003. Features of each dataset are described in Table 2. Here, the feature Max Slope is the highest immediate change rate in the dataset, and is an absolute value, meaning it can be a positive or negative slope. The first scenario, is meant to evaluate Timeliness. The Second Scenario, is meant to evaluate Latencies and Energy Management.

4.1 Settings and Results of the First Scenario

This Scenario evaluates Timeliness, and only involves the Application Layer. In this test, a node that schedules sensing intervals with the APP of POLA, is compared to a PSS Application Layer. Data is sensed by Application Layer. Original data (which the node is supposed to sense in the most accurate way), is recorded separately. Scheduling of sensing intervals in POLA depends only on data change rates, and since PSS has no elements of randomness, this test is deterministic, and no repetitions are required on each run.

Each run of this test, is performed in 48 **simulated units**, which can be treated as any time unit. All three datasets are evaluated on the same parameter space. Some values are expected to work differently on each dataset, because each dataset has different change rates and distributions of values. POLA requires five parameters for this test: Data Buffer size, with values 3,5,10,15,20; Step size, from 0.1 to 0.5, in steps of 0.1; Higher threshold, from 1 to 4, in steps of 1; Lower Threshold, from 0.2 to 1, in steps of 0.2; and Initial Sensing Rate, from 0.1 to 1.1, in steps of 0.2. The range for the Data Generator Period is from 0.1 to 1.1, in steps of 0.2. Such space generates 18.000 runs. In order to compare the APP of POLA, with the APP of PSS, applicable parameters are used with the same ranges: Initial Sensing Rate (it becomes the only sensing rate in Periodic Scheduling), and Data Generator Period. PSS requires 36 runs, because each of those parameters has six different values.

After applying all Timeliness metrics, near-median behaviours in POLA made a better use of oversampling, unlike near-median behaviours of PSS. Figure 1 shows how oversampling looks in near-median runs of POLA, for the Temperature Dataset. Notice that changes in sample densities can be observed for the POLA strategy, depending on change rates of nearby data, while on PSS, sample density is unique for each run. Therefore, POLA is able to adapt its sensing rates to change rates of data, reducing the energy and computational resources wasted in a PPS strategy.



(a) Temperature, POLA

Fig. 1. Adaptive use of Oversampling in a Near-Median run (POLA). Gray: Generated Data. Black: Sensed Data. Temperature Dataset.

Table 3. Median values and Quartile Ranges of $Hamming$ and N_{Trap} (POLA)

Dataset	$Hamming$	Quartile rng	N_{Trap}	Quartile rng
Wind Speeds	50	30 - 140	0.055	0.02 - 0.12
Temperature	25	-12 - 99	0.02	0.008 - 0.038
Artificial	43	17 - 75	0.015	0.005 - 0.038

Near-median values of $Hamming$ and N_{Trap} are in Table 3. In $Hamming$, the best Median case was obtained on the Temperature dataset, were the median of the Hamming Distance is of about 25 samples; since the dataset has 517 samples, this means that the average Hamming distance, is smaller than 6 percent of all samples. In N_{Trap}, the best case was obtained on the Artificially Generated Dataset, were the median value of the N_{Trap} was smaller than 0.02 percent of the dataset, followed by the Temperature Dataset, with a similar value. Values obtained on the Wind Speeds dataset also had a small median value, of 0.05 percent, however the range of the quartiles spreaded until 0.12 percent.

In the Histogram of the $AST_{shifting}$, values are better when they are closer to the zero of the horizontal axis. Distributions are better if they are not scattered, and have more near-zero values. The best case for this metric, was in the Wind Speeds Dataset, in Figure 2. The highest value was the second closest to zero. Moreover, around half of the $AST_{shifting}$, were smaller than a tenth of simulation time.

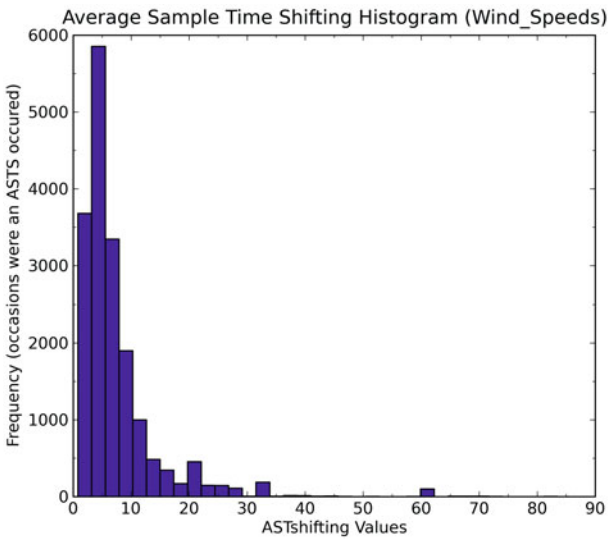


Fig. 2. Distribution of $AST_{shifting}$, WindSpeeds Dataset

4.2 Settings and Results of the Second Test Scenario

In the Second Test Scenario, a static network of 20 nodes is built and run for 48 time units. Performance metrics for Latencies and Energy Management are applied. This is repeated for each parameter. In SF, data is sent to the neighbour that first acknowledges a ping in order to forward it to a Sink node.

The POLA strategy for NWK is non-deterministic, as well as PSS with SF. The parameter space from the previous section implies an enormous number of runs, many of which are not representative (far from the median). Then, the need of building a representative parameter space, had risen. A subset from the parameter space of the previous tests was selected, based on parameters whose values overlap on the median values of all precision performance metrics. Such values are shown in Table 4.

For Latency metrics, the Median was chosen to represent the HCM, and the Mode was chosen to represent the MHC. The median of the HCM for POLA was of 1 Hop, while in PSS with SF, was of 2 hops. The mode for MHC in POLA was of 4 hops, while in PSS with SF, was of 17 hops for the Temperature dataset, and of over 40 hops for the rest.

Table 4. Parameter Values for the POLA strategy, used in Test 2

Dataset	DataBuffSize	StpSize	Thr_h	Thr_L	InitSensRate
Wind Speeds	20	0.5-0.9 stp 0.2	1-4 stp 1	0.2-1 stp 0.2	0.5
Temperature	15,20	0.5	1-4 stp 1	0.2-1 stp 0.2	0.7
Artificial	3,15	0.1-0.5 stp 0.2	1-4 stp 1	0.4, 0.6, 0.8	0.3-1.1 stp 0.4

Table 5. Median Overall Communication Energy Expenditure (OCEe), All Datasets, Test 2

Dataset	MedianOCEe(POLA)	MedianOCEe (PerSch with Simplif FW)
Wind Speeds	2635	5050
Temperature	1790	2450
Artificial	3450	3200

For Energy Management metrics, results of the OCEe for the POLA strategy and PSS with SF, on all three datasets, are shown in Table 5. The Median of OCEe, holds information on the most expectable case of Energy Expenditures. In the case of Table OCEe, the POLA strategy reached smaller consumptions than PSS with SF. In the case of the Wind Speeds dataset, such consumptions in POLA were near half the consumptions of the other strategy. In the Artificially Generated dataset, PSS with SF achieved slightly smaller consumptions than POLA.

As for the Means of the ICEe Vector, for the Windspeeds Dataset, even the highest mean values for ICEe, are below the values of PSS with SF, and ranges do not overlap at all. This phenomena is repeated for the Temperature Dataset, except for the small mean value at 340 of the POLA Histogram of this dataset, however, the most significant value of the Histogram is located below the range of PSS with SF. As for the Variances of the ICEe, the Dataset with the most evenly distributed consumption values in the network, is the Temperature dataset under the POLA strategy, followed by the Wind Speeds Dataset, in POLA, again. It can be noticed that on all cases, the POLA strategy achieved a more even distribution of energy consumption in the network, even on the Artificially Generated dataset. Such feature helps the network keep a steady number of nodes for longer times.

5 Conclusions and Future Work

In this paper, a data-centred scheduling strategy (POLA), where the WSN was modelled as an Asynchronous Cellular Automaton, was presented, as well as some performance metrics for Energy Management, Latencies and Timeliness.

Our results show that POLA performs well in Precision Metrics, even when Parameter Settings are not suited in the best possible values. Moreover, Periodic Scheduling used too much oversampling to satisfy performance metrics, while POLA adapted to change rates of data, preventing excessive oversampling from happening, and satisfying Precision metrics.

For Energy Management, POLA maintained not only a lower OCEe on most datasets, but on all cases, a more evenly distributed energy consumptions on the nodes of the network. As for Latencies, regardless of the used metric, POLA outperformed Periodic Scheduling with Simplified Forwarding, especially on the MHC, where the difference got to be of more than ten times the value.

Our future work will concentrate in incorporating mobility, making some improvements with the help of evolutionary techniques and artificial curiosity models, and focusing on rescue applications.

References

1. Esnaashari, M., Meybodi, M.R.: A cellular learning automata based clustering algorithm for wireless sensor networks. Sensor Letters - International Frequency Sensor Association (IFSA) 6(5), 723–735 (2008) ISSN 1546-198X
2. Li, W., Zomaya, A.Y., Al-Jumaily, A.: Cellular automata based models of wireless sensor networks. In: Proceedings of the 7th ACM International Symposium on Mobility Management and Wireless Access (2009)
3. Floreano, D., Mattiussi, C.: Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies. ACM Press, New York (2008) ISBN 0262062712
4. Anastasi, G., Conti, M., Francesco, M.D., Passarella, A.: Energy conservation in wireless sensor networks: A survey. Ad Hoc Networks 7, 537–568 (2009)
5. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Ad-Hoc Networks 3, 325–349 (2005)
6. Lin, J., Song, C., Wang, H.: A rule-based method for improving adaptability in pervasive systems. The Computer Journal, 1–14 (2007)
7. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks MobiCom (1999)
8. Cortez, P., Morais, A.: A data mining approach to predict forest fires using meteorological data. In: New Trends in Artificial Intelligence, Proceedings of the 13th EPIA - Portuguese Conference on Artificial Intelligence, pp. 512–523 (2007)

Using Evolutionary Neural Networks to Test the Influence of the Choice of Numeraire on Financial Time Series Modeling

Antonia Azzini, Mauro Dragoni, and Andrea G.B. Tettamanzi

Universita' degli Studi di Milano, Dipartimento di Tecnologie dell'Informazione
{antonia.azzini,mauro.dragoni,andrea.tettamanzi}@unimi.it

Abstract. This work presents an evolutionary solution that aims to test the influence of the choice of numeraire on financial time series modeling. In particular, the method used in such a problem is to apply a very powerful natural computing analysis tool, namely evolutionary neural networks, based on the joint evolution of the topology and the connection weights together with a novel similarity-based crossover, to a couple of very liquid financial time series expressed in their trading currency and several alternative numeraires like gold, silver, and a currency like the euro, which is intended to be stable 'by design', and compare the results.

1 Introduction

Many successful applications of natural computing techniques to the modeling and prediction of financial time series have been reported in the literature [5,6,7].

To our knowledge, the vast majority, if not the totality of such approaches, as well as of the modeling approached based on more traditional methods, consider the financial time series under investigation expressed in the currency it is usually traded against, e.g., US stock indices in US dollars, the FTSE 100 in British Pounds, the Continental European stock indices in euros, the Nikkei in yens, crude oil and most other commodities in dollars, and so on and so forth. In other word, the implicit assumption of the natural trading currency as the numeraire for the time series is quietly made, without even considering alternative choices.

However, recently, Turk [10], among others, has observed that, e.g., the price of crude oil, expressed in gold units, has remained essentially unchanged since the end of World War II, even though its price in US dollars (against which it is generally traded) has varied wildly since the dollar abandoned the gold parity. This suggests that precious metals, especially those historically used for monetary purposes, like gold and silver, may serve better than *fiat* currencies do as a numeraire.

In particular, the truth Turk's observation unveils that the US dollar, or any other *fiat* currency of choice, does not measure any absolute unit of value. This is in part reflected by its ever-changing purchase power, although the definition of the purchase power of a currency must be based on the more or less arbitrary selection of a basket of goods, and the values of the goods in any basket may

© Springer-Verlag Berlin Heidelberg 2011

oscillate due to the law of demand and supply. The fact is, by definition, that a unit of a *fiat* currency is an abstract promise to pay issued by a central bank, without even specifying what should be paid in exchange for that unit; therefore, its value depends on the reliability, reputation, and solvability of the issuer, which may vary at any time depending on the economical, political, and social situation.

If one accepts the above argument, a direct consequence is that, when analyzing a financial time series expressed in its trading currency, one is in fact analyzing the combined effect of changes in price of the security under investigation and of changes in the value of the currency used as numeraire, which might partially or completely obfuscate the patterns that are sought for.

One might try to use a basket of currencies as the numeraire for expressing a financial time series, but all currencies one might select are subject to the same oscillations. Nor is the use of official deflators a satisfactory replacement: as a matter of fact, there is no guarantee that the periodic revisions of the baskets used by census bureaus do not serve the agendas of the governments that, directly or indirectly, control them, and, at any rate, deflators only reflect the long-term tendency, averaged on a yearly basis, not the day-by-day fluctuations.

The basic research question we want to investigate in this paper is

> Does the numeraire used to express a financial time series have any influence on its predictability?

The method by which we will try to answer this research question is to apply a very powerful natural computing analysis tool, namely evolutionary neural networks, to a couple of very liquid financial time series expressed in their trading currency and several alternative numeraires like gold, silver, and a currency like the euro, which is intended to be stable "by design", and compare the results. The hypothesis we are seeking to accept or reject may be formulated as follows:

> A financial time series is more predictable if it is expressed in a numeraire whose value is more stable (ideally constant) in time.

Why use gold, silver, and euros as alternative numeraires? The case for gold and silver is twofold: on one hand, gold and silver have served as money from the dawn of history until the collapse of the gold standard in the last century; still today, gold is held as a reserve good by the most prominent central banks and used by private investors as a hedge against financial downturns; on the other hand, the supply of these two precious metals is extremely stable and as inelastic as it can be; a research by the World Gold Council [9] found that the purchasing power of gold remained stable in five different countries over extremely long timespans; therefore, they appear as two excellent candidates for the purpose of measuring value. As for the euro, we decided to consider it for many reasons: first of all, the euro was designed and is managed by the European Central Bank having stability as the primary goal; second, it is chasing the dollar as the alternative international trade and reserve currency; third, it is a currency and not a commodity like gold and silver; fourth, it is the currency of many Europeans, and its use looks appropriate for a paper sent to a European workshop.

This paper is organized as follows: Section 2 presents the problem and the technical indicators used to define the dataset, while a brief description of the evolutionary approach considered in this work is reported in Section 3. The results obtained from the experiments carried out by applying the evolutionary algorithm are presented in Section 4, together with a discussion of the performances obtained. Finally, Section 5 provides some concluding remarks.

2 Problem Description

The modeling problem we employ to test our hypothesis on the influence of the numeraire on the predictability of a financial time series is the most straightforward problem, consisting of predicting the amount of price variation of a financial instrument in the next period t, given the financial instrument's past history up to period i. In particular, the amount of price variation will be expressed in the form of a log-return $r_{i+1} = \log \frac{x_{i+1}}{x_i}$.

This prediction problem may be naturally regarded as an optimization problem, where we wish to minimize the mean square error of a multilayer perceptron whose input neurons provide information about the financial instrument's past history up to period i and whose output neuron provides an estimate of r_{i+1}.

To this aim, the input values for the predictor are given by the values of 11 technical indicators in period i for the financial instrument considered, corresponding to some of the most popular indicators used in technical analysis.

There are several different technical indicators used by practitioners, and the choice of those selected for this approach is made, at the beginning, also by selecting, together with the most widely used, the most representative of different categories (e.g. moving averages, oscillators) [8]. The rationale for this choice is that these indicators summarize important features of the time series of the financial instrument considered, and they represent useful statistics and technical information that otherwise should be calculated by each individual of the population, during the evolutionary process, increasing the computational cost of the entire algorithm. The list of all the inputs provided to neural network predictors is shown in Table 1.

Generally, technical indicators can be directly incorporated as model inputs, or, alternatively, they can be preprocessed to produce an input by taking ratios or through the use of rules. The last case is a combinatorial problem and traditional modeling methods can provide an infinite number of possibilities, in some cases problematic. This suggests that an evolutionary algorithm in which the model structure and model inputs are not defined a priori will have potential for generating trading operations drawn from individual technical indicators [4].

One target value is then defined for each day i of the time series considered with a value defined by Equation 1, representing the price variation of the considered financial index at the next day:

$$\text{target}_i = \ln \frac{\text{ClosingPrice}(i+1)}{\text{ClosingPrice}(i)}; \tag{1}$$

Table 1. Input Technical Indicators

Index	Input Tech. Ind.	Description
1	MA5(i)/Closing Price(i)	5-Day Moving Avg to Closing Price Ratio
2	EMA5(i)/Closing Price(i)	5-Day Exp. Moving Avg to Closing Price Ratio
3	MA20(i)/Closing Price(i)	20-Day Moving Avg to Closing Price Ratio
4	EMA20(i)/Closing Price(i)	20-Day Exp. Moving Avg to Closing Price Ratio
5	MA200(i)/Closing Price(i)	200-Day Moving Avg to Closing Price Ratio
6	EMA200(i)/Closing Price(i))	200-Day Exp. Moving Avg to Closing Price Ratio
7	MACD(i)	Moving Avg Conv./Div.
8	SIGNAL(i)	Exp. Moving Avg on MACD
9	Momentum(i)	Rate of price change
10	ROC(i)	Rate Of Change
11	RSI(i)	Relative Strength Index

All the data are preprocessed by applying a gaussian distribution with mean equal to 0 and standard deviation equal to 1.

3 The Neuro Genetic Algorithm

The overall algorithm is based on the evolution of a population of individuals, represented by Multilayer Perceptrons neural networks (MLPs), through a joint optimization of their structures and weights, here briefly summarized; a more complete and detailed description can be found in the literature [3]. The algorithm 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 applying BP 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 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.

3.1 Evolutionary Process

The initial population is randomly created and the genetic operators are then applied to each network until the termination conditions are not satisfied.

At each generation, the first half of the population corresponds to the best $\lfloor n/2 \rfloor$ individuals selected by truncation from a population of size n, while the second half of the population is replaced by the offsprings generated through the crossover operator. Crossover is then applied to two individuals selected from the best half of the population (parents), with a probability parameter p_{cross}, defined by the user together with all the other genetic parameters, and maintained unchanged during the entire evolutionary process.

It is worth noting that the p_{cross} parameter refers to a 'desired' crossover probability, set at the beginning of the evolutionary process. However, the 'actual' probability during a run will usually be lower, because the application of the crossover operator is subject to the condition of similarity between the parents.

Elitism allows the survival of the best individual unchanged into the next generation and the solutions to get better over time. Then, 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.

The application of the genetic operators to each network is described by the following pseudo-code:

```
1. Select from the population (of size $n$) $\lfloor n/2 \rfloor$ individuals by truncation and
   create a new population of size $n$ with copies of the selected individuals.
2. For all individuals in the population:
   (a) Randomly choose two individuals as possible parents.
   (b) Check their local similarity and apply crossover according to the crossover
       probability.
   (c) Mutate the weights and the topology of the offspring according to the
       mutation probabilities.
   (d) Train the resulting network using the training set.
   (e) Calculate the fitness $f$ on the test set.
   (f) Save the individual with lowest $f$ as the best-so-far individual if the $f$
       of the previously saved best-so-far individual is higher (worse).
3. Save statistics.
```

The `SimBa` crossover starts by looking for a 'local similarity' between two individuals selected from the population. If such a condition is satisfied the layers involved in the crossover operator are defined. The contribution of each neuron of the layer selected for the crossover is computed, and the neurons of each layer are reordered according to their contribution. Then, each neuron of the layer in the first selected individual is associated with the most 'similar' neuron of the layer in the other individual, and the neurons of the layer of the second individual are re-ranked by considering the associations with the neurons of the first one. Finally a cut-point is randomly selected and the neurons above the cut-point are swapped by generating the offspring of the selected individuals.

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 algorithms. 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, indicated as p^+_{layer}, p^-_{layer} and p^+_{neuron}, 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.

For each generation of the population, all the information of the best individual is saved.

As previously considered [2,1], 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 used for evaluating each individual in the population is proportional to the mean square error (mse) and to the computational cost of the considered network. This latter term induces a selective pressure favoring individuals with reduced-dimension topologies.

The fitness function is calculated, after the training and the evaluation processes, by the Equation 2

$$f = \lambda kc + (1 - \lambda) * mse, \tag{2}$$

where λ corresponds to the desired tradeoff between network cost and accuracy, and it has been set experimentally to 0.2 to place more emphasis on accuracy, since the NN cost increase is checked also by the entire evolutionary algorithm. k is a scaling constant set experimentally to 10^{-6}, and c models the computational cost of a neural network, proportional to the number of hidden neurons and synapses of the neural network.

Following the commonly accepted practice of machine learning, the problem data is partitioned into training, test and validation sets, used, respectively for network training, to stop learning avoiding overfitting, and to test the generalization capabilities of a network. The fitness is calculated over the test set.

4 Experiments and Results

We selected two of the most liquid US market indices, namely the Standard & Poors 500 (SP500) and the Dow Jones Industrial Average (DJIA), taking into account the daily closing prices of the time series of the last decade (from October 2000 to October 2010). All data are divided into three subsets, according to the percentages of 50%, 25% and 25%, respectively, for the training, test, and validation set. The latter corresponds to the most recent data, while the oldest are used to define the training set.

Four versions of the datasets for either of these two financial indices are considered, obtained by expressing the prices in four distinct numeraires: besides the US dollar, their usual currency, we also used the euro and two precious metals already used in the past for monetary purposes, namely gold and silver. For each of the numeraires considered, a set of experiments has been carried out, and the performances of the models obtained have been compared.

All the experiments consider the same parameters setting used in previous work [1], that produced the optimal average accuracies. In particular, the topology parameters p^+_{layer}, p^-_{layer}, and p^+_{neuron} have all been set to 0.05, while the crossover parameter p_{cross} has been set to 0.7.

We performed 20 runs for each experiment, with 40 generations and a population size of 60 individuals for each run. The number of epochs used to train the neural network represented by each individual by BP is 250.

We carried out two groups of experiments: the first is carried out by considering the overall dataset for training, test, and validation, while the second is carried out by considering, separately, the two halves of each dataset. Each of the two halves of the entire dataset covers a period of about five years: the most recent one spans the period from October 2005 (the second half) to October 2010, while the oldest one refers to the period from October 2000 to October 2005 (the first half).

4.1 Results

The results of the first group of experiments are reported in Table 2, while those of the second group of experiments are reported in Tables 4 and 6. All the tables show the errors obtained, respectively, on the DJIA and SP500 datasets: the second and third columns contain the error reported by the best individual over the 20 runs, and the difference w.r.t. the USD, which is used as a baseline. The fourth and the fifth columns contain the same data above, but referred to the average error obtained by the best individuals of each run; finally, the last column contains the standard deviation of the average error. To make it easier to read the tables, both the errors and the standard deviations have been multiplied by 10^3.

To test the significance of the results we applied Student's t-test. Table 3 shows the significance levels for all the numeraire combinations for both datasets on the first group of experiments, while Tables 5 and 7 show the significance levels obtained on the second group.

From Table 2, we can notice that, for both indices, the use of the three alternative numeraires leads to models that outperform those that consider USD. The improvement is particularly striking with silver and gold. Such performance is closely followed by EUR. We can also notice that the adoption of gold and silver as numeraire leads to more robust solutions.

On the SP500 dataset, the best performance is obtained by using silver as numeraire.

Table 2. Errors on the DJIA and SP500 indices over the entire dataset

Dataset	Best Individual		Average		Standard Deviation
	Error	Diff. w.r.t. USD	Error	Diff. w.r.t. USD	
DJIA USD	4.16	-	4.20	-	0.00376
DJIA EUR	2.46	- 40.87%	2.75	- 34.52%	0.00948
DJIA SILVER	1.65	- 60.34%	1.66	- 60.48%	0.00100
DJIA GOLD	1.17	- 71.88%	1.17	- 72.14%	0.00009
SP500 USD	4.28	-	4.29	-	0.00067
SP500 EUR	2.65	- 38.08%	2.92	- 31.93%	0.01530
SP500 SILVER	1.53	- 64.25%	1.53	- 64.34%	0.00006
SP500 GOLD	1.84	- 57.01%	1.86	- 56.64%	0.00174

Table 3. Percentage of significance level of Student's t-test over the entire dataset

	DowJones Index	SP500 Index
USD vs. EUR	90.95%	85.66%
USD vs. SILVER	99.99%	100.00%
USD vs. GOLD	99.99%	99.99%
EUR vs. SILVER	84.96%	86.96%
EUR vs. GOLD	96.80%	72.99%
SILVER vs. GOLD	95.22%	71.00%

Table 4. Errors on the DJIA and SP500 indices over the recent half dataset

Dataset	Best Individual		Average		Standard Deviation
	Error	Diff. w.r.t. USD	Error	Diff. w.r.t. USD	
DJIA USD	0.58	-	0.70	-	0.00617
DJIA EUR	0.37	- 36.21%	0.39	- 44.29%	0.00153
DJIA SILVER	1.54	+ 165.52%	1.71	+ 144.29%	0.00496
DJIA GOLD	0.29	- 50.00%	0.30	- 57.14%	0.00388
SP500 USD	0.66	-	0.81	-	0.00789
SP500 EUR	1.68	+ 154.55%	1.71	+ 111.11%	0.00149
SP500 SILVER	0.46	- 30.30%	0.48	- 40.74%	0.00246
SP500 GOLD	0.38	- 42.42%	0.40	- 50.62%	0.00050

Table 5. Percentage of significance level of Student's t-test over the recent half dataset

	DowJones Index	SP500 Index
USD vs. EUR	37.09%	79.00%
USD vs. SILVER	80.47%	34.36%
USD vs. GOLD	41.35%	46.79%
EUR vs. SILVER	97.01%	98.95%
EUR vs. GOLD	13.15%	99.98%
SILVER vs. GOLD	95.41%	17.52%

Table 6. Errors on the DJIA and SP500 indices over the oldest half dataset

Dataset	Best Individual		Average		Standard Deviation
	Error	Diff. w.r.t. USD	Error	Diff. w.r.t. USD	
DJIA USD	1.03	-	1.04	-	0.00143
DJIA EUR	0.25	- 75.73%	0.26	- 75.00%	0.00039
DJIA SILVER	0.34	- 66.99%	0.34	- 67.31%	0.00012
DJIA GOLD	0.12	- 88.35%	0.12	- 88.46%	0.00002
SP500 USD	1.00	-	1.01	-	0.00073
SP500 EUR	0.23	- 77.00%	0.28	- 72.28%	0.00451
SP500 SILVER	0.36	- 64.00%	0.37	- 63.37%	0.00021
SP500 GOLD	0.14	- 86.00%	0.16	- 84.16%	0.00150

The t-test on the results of the first group of experiments shows a very high significance for all combinations that include the USD baseline, except for DJIA-USD vs. DJIA-EUR, whose significance level is about 86%.

The results of the second group of experiments confirm those obtained on the first group, except for the silver numeraire on the half recent dataset of DJIA, and by the EUR numeraire on the half recent dataset of SP500. However, for both datasets, the solutions found are more robust than the USD baseline.

Also the percentage of significance level of the t-test over the second group of experiments produces similar results to those obtained from the complete

Table 7. Percentage of significance level of Student's *t*-test over the oldest half dataset

	DowJones Index	SP500 Index
USD vs. EUR	98.46%	82.68%
USD vs. SILVER	98.07%	99.35%
USD vs. GOLD	99.80%	98.19%
EUR vs. SILVER	39.63%	14.08%
EUR vs. GOLD	62.17%	16.37%
SILVER vs. GOLD	98.06%	50.35%

datasets. Indeed, for all numeraires, the errors found are smaller than the USD baseline, for both DJIA and SP500. Furthermore, while the improvements of performance obtained on the most recent halves are only marginally significant (see Table 5), the improvements obtained over the recent datasets are highly significant (see Table 7).

4.2 Discussion

At first sight, the results look very interesting, as they appear to suggest that, with a few exceptions, the indices considered become more predictable if expressed in an alternative numeraire.

If one takes the significance tests into account, though, conclusions are to be drawn much more carefully. On the entire dataset, there are two clear winners, namely gold and silver, and one clear loser, the US dollar. The euro is only marginally better than the US dollar and marginally worse than the two precious metals. On the most recent five-year dataset, there is no statistically significant winner among gold, the euro, and the US dollar, but there is a clear loser, silver, whose performance is disastrous. On the least recent five-year dataset, there appears to be a clear ranking of numeraires, namely gold > euro > silver > US dollar, although the significance test tells us that the only statistically significant difference is between the two precious metals and the US dollar.

Therefore, if we were to use these results to select one numeraire for use in the analysis of financial time series, there would be just one sensible and consistent choice, namely to use gold. This is the only numeraire that would have consistently given significantly better results than the US dollar (the natural numeraire) for both instruments and for all three time-frames considered.

5 Conclusion and Future Work

We asked whether the numeraire used to express a financial time series has any influence on its predictability, and we attempted to answer this question by applying a very powerful natural computing analysis tool to a couple of very liquid financial time series expressed in their trading currency and several alternative numeraires.

The experimental evidence suggests that the hypothesis that a financial time series is more predictable if it is expressed in a numeraire whose value is more stable is to be accepted, if only in the perhaps less ambitious formulation that indicates gold, in particular, as the alternative numeraire.

One may object, with reason, that two financial time series, albeit probably among the most liquid of the world, are not a sufficient base for drawing such a general conclusion. As a matter of fact, more evidence is needed to turn such a preliminary indication into an established fact. Therefore, a possible extension of our work would be to test this hypothesis on a larger sample of financial instruments, representative of the markets of various regions of the globe.

References

1. Azzini, A., Dragoni, M., Tettamanzi, A.: A novel similarity-based crossover for artificial neural network evolution. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 344–353. Springer, Heidelberg (2010)
2. Azzini, A., Tettamanzi, A.: Evolving neural networks for static single-position automated trading. Journal of Artificial Evolution and Applications (Article ID 184286), 1–17 (2008)
3. Azzini, A., Tettamanzi, A.: A new genetic approach for neural network design. In: Engineering Evolutionary Intelligent Systems. SCI, vol. 82. Springer, Heidelberg (2008)
4. Brabazon, A., O'Neill, M.: Biologically Inspired Algorithms for Financial Modelling. Springer, Berlin (2006)
5. Brabazon, A., O'Neill, M. (eds.): Natural Computing in Computational Finance. SCI, vol. 1. Springer, Heidelberg (2008)
6. Brabazon, A., O'Neill, M. (eds.): Natural Computing in Computational Finance. SCI, vol. 2. Springer, Heidelberg (2009)
7. Brabazon, A., O'Neill, M., Maringer, D. (eds.): Natural Computing in Computational Finance. SCI, vol. 3. Springer, Heidelberg (2010)
8. Colby, R.: The Encyclopedia Of Technical Market Indicators, 2nd edn. McGraw-Hill, New York (2002)
9. Harmston, S.: Gold as a store of value. Tech. Rep. 22, World Gold Council, London (November 1998)
10. Turk, J.: The barbarous relic — it is not what you think. Monograph 55, Committee for Monetary Research and Education, Inc. (January 2006)

Market Microstructure: Can Dinosaurs Return? A Self-Organizing Map Approach under an Evolutionary Framework

Michael Kampouridis[1], Shu-Heng Chen[2], and Edward Tsang[1]

[1] School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, CO4 3SQ, UK
[2] AI-Econ Center, Department of Economics, National Cheng Chi University, Taipei, Taiwan 11623

Abstract. This paper extends a previous market microstructure model, which investigated fraction dynamics of trading strategies. Our model consisted of two parts: Genetic Programming, which acted as an inference engine for trading rules, and Self-Organizing Maps (SOM), which was used for clustering the above rules into trading strategy types. However, for the purposes of the experiments of our previous work, we needed to make the assumption that SOM maps, and thus strategy types, remained the same over time. Nevertheless, this assumption could be considered as strict, and even unrealistic. In this paper, we relax this assumption. This offers a significant extension to our model, because it makes it more realistic. In addition, this extension allows us to investigate the dynamics of market behavior. We are interested in examining whether financial markets' behavior is non-stationary, because this implies that strategies from the past cannot be applied to future time periods, unless they have co-evolved with the market. The results on an empirical financial market show that its behavior constantly changes; thus, agents' strategies need to continuously adapt to the changes taking place in the market, in order to remain effective.

Keywords: Genetic Programming, Self-Organizing Maps, Market Microstructure, Market Behavior.

1 Introduction

There are several types of models in the agent-based financial markets literature. One way of categorizing them is to divide them into the N-type models and the Santa-Fe Institute (SFI) like ones [2]. The former type of models focuses on the mesoscopic level of markets, by allowing agents to choose among different types of strategies. A typical example is the fundamentalist-chartist model. Agents in this model are presented with these two strategy types and at any given time they have to choose between these two. A typical area of investigation of these models is fraction dynamics, i.e., how the fractions of the different strategy types change over time. However, what is not presented in most of these models is

© Springer-Verlag Berlin Heidelberg 2011

novelty-discovering agents. For instance, in the fundamentalist-chartists example, agents can only choose between these two types; they cannot create new strategies that do not fall into either of these types. On the other hand, the SFI-like models overcome this problem by focusing on the microscopic level of the markets. By using tools such as Genetic Programming [7], these models allow the creation and evolution of novel agents, which are not constrained by pre-specified strategy types.[1] However, this kind of models tends to focus on price dynamics, rather than fraction dynamics [2].

In a previous work [3], we combined properties from the N-type and SFI-like models into a novel model. We first used Genetic Programming (GP) as a rule inference engine, which created and evolved autonomous agents; we then used Self-Organizing Maps (SOM) [6] as a clustering machine, and thus re-created the mesoscopic level that the N-type models represent, where agents were categorized into different strategy types. We then investigated the short- and long-term dynamics of the fractions of strategies that existed in a financial market. Nevertheless, that study rested upon an important assumption, i.e., the maps derived from each time period were comparable with each other. This comparability assumption itself required that the types (clusters), as well as their operational specification, would not change over time. If this were not the case, the subsequent study would be questioned. This was mainly due to one technical step in our analysis called translation. The purpose of translation was to place the behavior of agents observed in one period into a different period and to recluster it for the further cross-period comparison. We could not meaningfully have done this without something like topological equivalence, which could not be sustained without the constancy of the types.

However, this assumption can be considered as strict and unrealistic. Strategy types do not necessarily remain the same over time. For instance, if a chartist strategy type exists in time t, it is not certain it will also exist in $t+1$. If market conditions change dramatically, the agents might consider other strategy types as more effective and choose them. The chartist strategy would then stop existing.

In this paper, we relax the above assumption, since our current work does not require cross-period comparisons. Our model thus becomes more realistic. In addition, *we shift our focus from fraction dynamics to behavior dynamics*: we examine the plausibility of an observation made under artificial markets [1], which suggests that the nature of financial markets constantly changes. This implies that trading strategies need to constantly co-evolve with the markets; if they do not, they become obsolete or *dinosaurs* [1]. We hence test if this observation holds in the 'real' world, under an empirical financial market. This will offer important insights regarding the behavior dynamics of the markets.

The rest of this paper is organized as follows: Section 2 presents our model, and Sect. 3 briefly presents the GP algorithm we use. Section 4 then presents the experimental designs, Sect. 5 reviews the testing methodology, and Sect. 6 presents the results of our experiments. Finally, Sect. 7 concludes this paper.

[1] We refer the reader to [2], which provides a thorough review on both N-type and SFI-like models, along with a detailed list of them.

2 Model

2.1 Genetic Programming as a Rule-Inference Engine

The use of GP is motivated by considering the market as an evolutionary and selective process.[2] In this process, traders with different behavioral rules participate in the markets. Those behavioral rules which help traders gain lucrative profits will attract more traders to imitate, and rules which result in losses will attract fewer traders. An advantage of GP is that it does not rest upon any prespecified class of behavioral rules, like many other models in the agent-based finance literature [2]. Instead, in GP, a population of behavioral rules is randomly initiated, and the survival-of-the-fittest principle drives the entire population to become fitter and fitter in relation to the environment. In other words, given the non-trivial financial incentive from trading, traders are aggressively searching for the most profitable trading rules. Therefore, the rules that are outperformed will be replaced, and only those very competitive rules will be sustained in this highly competitive search process.

Hence, GP can help us infer what are the rules the traders follow, by simulating the evolution of the microstructure of the market. Traders can then be clustered based on realistic, and possibly complex behavioral rules.The GP algorithm used to infer the rules is presented in detail, later, in Sect. 3.

2.2 Self Organizing Maps for Clustering

Once a population of rules is inferred from GP, it is desirable to cluster them based on a chosen similarity criterion so as to provide a concise representation of the microstructure. The similarity criterion which we choose is based on the *observed trading behavior*.[3] Based on this criterion, two rules are similar if they are *observationally equivalent* or *similar*, or, alternatively put, they are similar if they generate the same or similar market timing behavior.[4]

Given the criterion above, the behavior of each trading rule can be represented by its series of market timing decisions over the entire trading horizon, for example, 6 months. Therefore, if we denote the decision "enter the market" by "1" and "leave the market" by "0", then the behavior of each rule is a binary vector. The dimensionality of these vectors is then determined by the length of the trading horizon. For example, if the trading horizon is 125 days long, then the dimension of the market timing vector is 125. Once each trading rule is concretized into its market timing vector, we can then easily cluster these rules by applying Kohonen's Self-Organizing Maps to the associated clusters.

[2] See [8] for his eloquent presentation of the *Adaptive Market Hypothesis.*

[3] Other similarity criteria could take place, too, such as risk averseness. However, in this paper we wanted to focus on the behavioral aspects of the rules.

[4] One might question the above similarity criterion, since very different rules might be able to produce the same signals. This does not pose a problem in this work, since we are interested in the behavior of the market (and thus the rules' behavior). We are not interested in the semantics aspect of the rules.

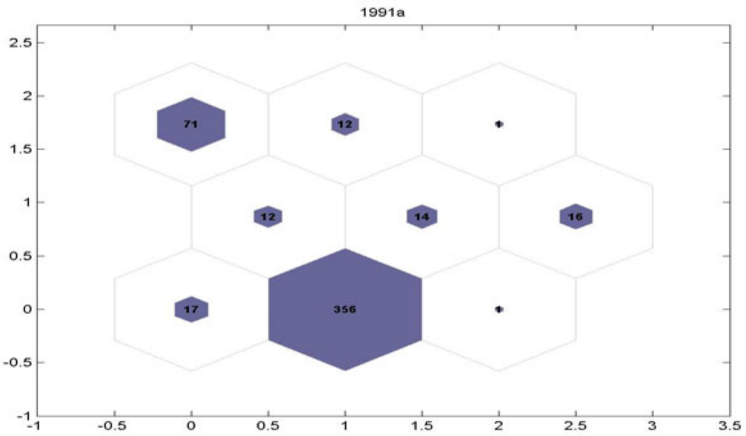


Fig. 1. Example of a 3 × 3 Self-Organizing Map

Figure 1 presents a 3×3 SOM. Here, 500 artificial traders are grouped into nine clusters. In a sense, this could be perceived as a snapshot of a nine-type agent-based financial market dynamics. Traders of the same type indicate that their market timing behavior is very similar. The market fraction or the size of each cluster can be seen from the number of traders belonging to that cluster. Thus, we can observe that the largest cluster has a market share of 71.2% (356/500), whereas the smallest one has a market share of 0.2% (1/500).

3 GP Algorithm

Our GP is inspired by a financial forecasting tool, EDDIE [4], which applies genetic programming to evolve a population of market-timing strategies, which guide investors on when to buy or hold. These market timing strategies are formulated as decision trees, which, when combined with the use of GP, are referred to as *Genetic Decision Trees* (GDTs). Our GP uses indicators commonly used in technical analysis: Moving Average (MA), Trader Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).[5] Each indicator has two different periods, a short- and a long-term one (12 and 50 days). Figure 2 presents a sample GDT generated by the GP.

Depending on the classification of the predictions, there are four cases: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). We then use the following 3 metrics, presented in Equations (1)-(3):

[5] We use these indicators because they have been proved to be quite useful in previous works like [4]. However, the purpose of this work is not to provide a list of the ultimate technical indicators.

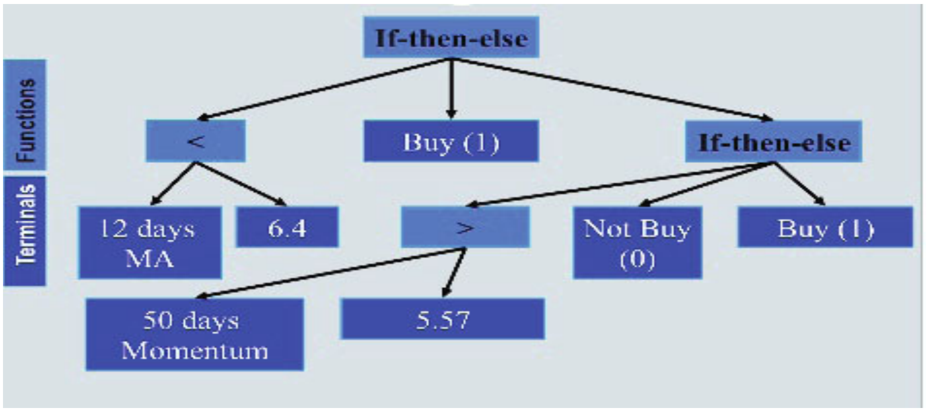


Fig. 2. Sample GDT generated by the GP

Rate of Correctness

$$RC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Rate of Missing Chances

$$RMC = \frac{FN}{FN + TP} \tag{2}$$

Rate of Failure

$$RF = \frac{FP}{FP + TP} \tag{3}$$

The above metrics combined give the following fitness function:

$$ff = w_1 * RC - w_2 * RMC - w_3 * RF \tag{4}$$

where w_1, w_2 and w_3 are the weights for RC, RMC and RF, respectively, and are given in order to reflect the preferences of investors. For instance, a conservative investor would want to avoid failure; thus a higher weight for RF should be used. For our experiments, we chose to include GDTs that mainly focus on correctness and reduced failure. Thus these weights have been set to 1, $\frac{1}{6}$ and $\frac{1}{2}$, respectively.

Given a set of historical data and the fitness function, GP is then applied to evolve the market-timing strategies in a standard way. After evolving a number of generations, what survives at the last generation is, presumably, a population of financial agents whose market-timing strategies are financially rather successful.

4 Experimental Designs

The experiments are conducted for a period of 17 years (1991-2007) and the data are taken from the daily closing prices of the market index of STI (Singapore). For statistical purposes, we repeat our experiments for 10 times.

Each year is split into 2 halves (January-June, July-December), so in total, out of the 17 years, we have 34 periods.[6] The first semester of a year is denoted with an 'a' at the end (e.g., 1991a), and the second semester of a year is denoted with a 'b' (e.g., 1991b). The GP systems is therefore executed 34 times, i.e., one time per period. Table 1 presents the GP parameters for our experiments. The GP parameters for our experiments are the ones used by Koza [7]. Only the tournament size has been lowered, because we were observing premature convergence. Other than that, the results seem to be insensitive to these parameters.

Table 1. GP Parameters

GP Parameters	
Max Initial Depth	6
Max Depth	17
Generations	50
Population size	500
Tournament size	2
Reproduction probability	0.1
Crossover probability	0.9
Mutation probability	0.01

After generating and evolving strategies for each one of the 34 periods, we then use SOM to cluster these strategies into types. We do this for every one of the 34 periods. Thus, we end up with 34 different SOMs, one per semester, which represent the market in different time periods over the 17-year horizon.

Finally, we define as 'base period', the period during which GP creates and evolves GDTs. We also define 'future period(s)', as the period(s) which follow(s) the base period (in chronological order).

5 Testing Methodology

In order to investigate whether the behavior of markets is non-stationary, we recluster the GDTs of each base period, to all future periods' clusters.[7] By applying the same GDTs (strategies) to clusters of future periods, we can observe how well these strategies fit in the new environments (clusters). The logic behind this is the following: when we first evolved and clustered the GDTs (base period), these GDTs were placed in clusters that represented their respective strategies. For instance, if there was a strategy type (cluster) that represented 'chartists', then all GDTs which followed a chartist strategy were placed in that cluster. When we then take the GDTs from a base period and recluster them to strategy

[6] At this point the length of the period is chosen arbitrarily as 6 months. We leave it to a future research to examine if and how this time horizon can affect our results.

[7] The process of reclustering is explained later in this section.

types of future periods, it is not guaranteed that there will again be a cluster that represents chartists. If the market constantly changes, there is a possibility that this type of strategies does not exist any more in the future periods. Thus, the GDTs find themselves *unadapted* to the new environment (clusters) and have to choose another cluster, which represents them as closely as possible. This cluster will be the one that has the centroid with the smallest Euclidean distance[8] from the market-timing vectors of these GDTs. Of course, since now the SOM of the future period is formed by different clusters, the GDTs might not fit in as well as they did in the base period. In order to measure this 'unfitting', we use a 'dissatisfaction rate', i.e., how dissatisfied these GDTs will be when placed into a future period's cluster that does not represent their strategy. *If the market is non-stationary, the GDTs' dissatisfaction rate will be high*, as a result of the changes that took place in the market. The dissatisfaction rate is defined as the Euclidean distance of a GDT's market-timing vector to the centroid of the cluster in which it is placed, after the reclustering procedure. Under a non-stationary market behavior, the following statement should hold:

The average dissatisfaction rate of the population of GDTs from future periods should not return to the range of dissatisfaction of the base period.

Hence, we will test the above statement against the STI index.

Let us now explain the process of reclustering. We start with 1991a as the base period. Each evolved GDT is moved to the next period, 1991b, and reclustered into one of the clusters of that period. In order to 'decide' which cluster to choose, the GDT compares the Euclidean distance of its market timing vector to the centroid of each cluster; it is then placed into the cluster with the smallest Euclidean distance. The same procedure follows for all GDTs of the population. At the end, the population of evolved GDTs from the base period of 1991a will have been reclustered into the clusters of period 1991b. The same procedure is followed in all future periods. This means that the GDTs from 1991a are also reclustered into 1992a, 1992b, ..., 2007b. Finally, the same process is done for all other base periods (i.e., 1991b, 1992a, ..., 2007a).

Once the process of reclustering is complete, we calculate the dissatisfaction rate of each GDT in the population. Next, we calculate the population's average dissatisfaction rate. We do the same for all 34 periods. Given a base period, the population average dissatisfaction of all periods is normalized by dividing those population average dissatisfaction rates by the population average dissatisfaction rate in the base period. Hence, each base period has its normalized average dissatisfaction rate equal to 1. In order to prove that the market is non-stationary, we need to show that the normalized average dissatisfaction rate of the GDTs increases in the future periods, and never returns to its initial value of 1, which was during the base period. If, on the other hand, this rate reaches 1 or below, *it is an indication of a cyclic market behavior*, since the GDTs have found the same conditions with the base period, and as a result feel as 'satisfied' as before.

[8] One may wonder if the choice of the Euclidean distance as a distance metric, when the vectors of the GDTs are binary, is an appropriate one. However, this does not pose a problem, because the vectors of the clusters' centroids are real valued.

Finally, we define as *dinosaurs* the population of GDTs that has been reclustered from a base period to future periods. The reason of calling them in this way is because these GDTs have not adapted to the new market environment (clusters of the SOMs from future periods) and are thus ineffective. If these GDTs' normalized average dissatisfaction rate drops to *less than or equal to 1*, we call them *returning dinosaurs*, because they have become effective again.[9]

6 Results

As explained, returning dinosaurs denote a cyclic market behavior. To examine if dinosaurs return, we iterate through each base period and calculate the minimum normalized average dissatisfaction rate for each future period. This gives us an indication of how many returning dinosaurs, if any, exist. If, for instance, 1991a is the base period, then there is a series of 33 population dissatisfaction values for its future periods. We obtain the minimum value among these 33 values, in order to check how close to 1 this future period is. This process is then repeated for 1991b and its 32 future periods, and so on, until base period 2007a. We thus end up with a 1×33 vector, which presents the minimum dissatisfaction per base period and thus shows whether any returning dinosaurs exist. In addition, we are interested in investigating whether different number of clusters (strategy types) can affect the test results. We thus run tests under 2 to 9 clusters, for the following SOM dimensions: 2×1, 3×1, 2×2, 5×1, 3×2, 7×1, 4×2, and 3×3. The graphs of the minimum dissatisfaction vectors for the STI index are presented in Fig. 3. Each line represents the results of a different SOM dimension. The horizontal line indicates a dissatisfaction of 1, and is given as a reference.

What we can see from Fig. 3 is that there are no base periods with a minimum normalized dissatisfaction rate below 1. In fact, the closest to 1 this rate gets is around 2 (1998a). The first row of Table 2 presents the average of the minimum dissatisfaction rate per cluster and verifies this observation. As we can see, the minimum dissatisfaction rate is on average 3.56 for the 2×1 SOM, and it gradually increases, as the number of clusters increases, reaching 5.79 for the 3×3 SOM. Hence, the minimum dissatisfaction rate is on average quite far away from 1, which as we mentioned is the threshold for a returning dinosaur.

In addition, the second row of Table 2 informs us that the average dissatisfaction rate per cluster is even higher, and ranges from 5.17 (2 clusters) to 8.88 (9 clusters). It is thus obvious that *on average, no dinosaurs return.* But even if we want to take into account the outliers (minimum dissatisfaction rate-Fig. 3 and Table 2), we can see that while the rate can get relatively low, it never reaches 1. This leads us to argue that *dinosaurs do not return or return only as lizards.* More specifically, the strategies (GDTs) found the new environments (clusters)

[9] In a previous work [5], where we investigated the markets' behavior dynamics by only using GP but not SOM, we did not use this 'strict' definition of returning dinosaurs. This led us to conclude that returning dinosaurs existed. However, if we had also used the current paper's definition, the results from [5] would not have dramatically differed from the current paper.

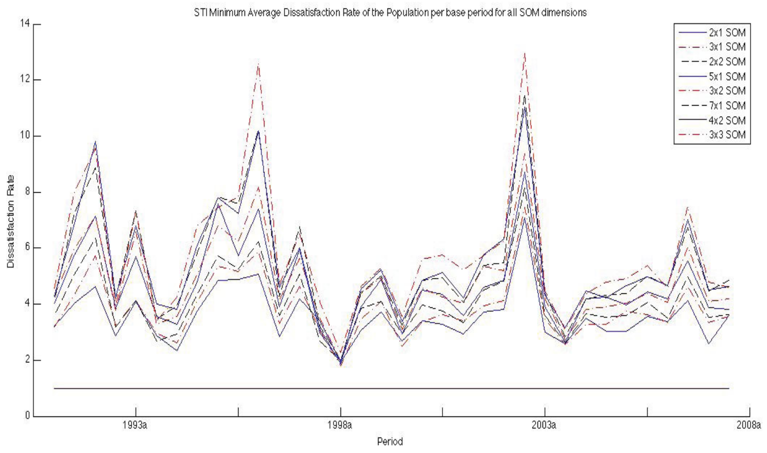


Fig. 3. Minimum normalized population dissatisfaction rate among all future periods for each base period for the STI index. Each line represents a different SOM dimension.

Table 2. Average Minimum Dissatisfaction (A.M.D.-row 1) and Average Dissatisfaction (A.D.-row 2) Rate per Cluster

	2×1	3×1	2×2	5×1	3×2	7×1	4×2	3×3
Mean of A.M.D.	3.56	3.83	4.09	4.61	4.79	5.34	5.41	5.79
Mean of A.D.	5.17	5.65	6.11	6.98	7.19	8.30	8.33	8.88

very different from the ones in their base period and were very 'dissatisfied'. The strategies that had not adapted to the market changes could not fit in the new environment. The above observation allows us to conclude that STI's behavior constantly changes. However, this behavior can sometimes resemble older ones. When this happens, old strategies might perform relatively well again (i.e., dinosaurs return as lizards). Nevertheless, strategies that have not co-evolved with the market, cannot reach performance levels as the ones they once had in their base period (i.e., no returning dinosaurs). Market conditions have changed and unless these strategies follow the changes, they become dinosaurs and thus ineffective.

One final observation we can make is that the number of clusters does not affect the test's results. The dissatisfaction rate of each market follows always the same pattern, regardless the number of clusters. No returning dinosaurs are observed, under any number of the trading strategy types tested.

7 Conclusion

To conclude, this paper presented a significant extension to a previous market microstructure model [3], and also discussed preliminary results on the behavior dynamics of financial markets. Our experimental work was inspired by an an observation made under artificial agent-based financial markets [1]. This observation says that the nature and constituents of agents, and thus their strategies, constantly change; if these strategies do not continuously adapt to the changes in their environments, then they become obsolete (dinosaurs). The results showed that on average, the dataset tested in this paper, STI (Singapore), did not demonstrate the existence of returning dinosaurs, and thus *verified the existence of the non-stationary property in financial markets' behavior.* The implications of this are very important. Strategies from the past cannot be successfully re-applied to future periods, unless they have co-evolved with the market. If they have not, they become obsolete, because the market conditions change continuously. They can occasionally return as lizards, meaning that these strategies can sometimes demonstrate relatively good performance, but they cannot become again as successful, as they once were. The next step of our research is to explore other markets and see if the above results are a universal phenomenon.

Acknowledgments

This work was supported by an EPSRC grant (No. EP/P563361/01) and by AI-Econ Center of National Cheng Chi University, Taiwan.

References

1. Arthur, B.: On learning and adaptation in the economy, working paper 92-07-038, Santa Fe Institute (1992)
2. Chen, S.H., Chang, C.L., Du, Y.R.: Agent-based economic models and econometrics. Journal of Knowledge Engineering Review (2010) (forthcoming)
3. Chen, S.H., Kampouridis, M., Tsang, E.: Microstructure dynamics and agent-based financial markets. In: Bosse, T., Geller, A., Jonker, C.M. (eds.) MABS XI 2010. LNCS (LNAI), vol. 6532, pp. 121–135. Springer, Heidelberg (2011)
4. Kampouridis, M., Tsang, E.: EDDIE for investment opportunities forecasting: Extending the search space of the GP. In: Proceedings of the IEEE Conference on Evolutionary Computation, Barcelona, Spain, pp. 2019–2026 (2010)
5. Kampouridis, M., Chen, S.H., Tsang, E.: Testing the dinosaur hypothesis under empirical datasets. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 199–208. Springer, Heidelberg (2010)
6. Kohonen, T.: Self-organized formation of topologically correct feature maps. Journal of Biological Cybernetics 43, 59–69 (1982)
7. Koza, J.: Genetic Programming: On the programming of computers by means of natural selection. MIT Press, Cambridge (1992)
8. Lo, A.: The adaptive market hypothesis: market efficiency from an evolutionary perspective. Journal of Portfolio Management 30, 15–29 (2004)

Macro-economic Time Series Modeling and Interaction Networks

Gabriel Kronberger[1], Stefan Fink[2],
Michael Kommenda[1], and Michael Affenzeller[1]

[1] Upper Austria University of Applied Sciences,
Research Center Hagenberg,
Softwarepark 11, 4232 Hagenberg, Austria
{gabriel.kronberger,michael.kommenda,michael.affenzeller}@fh-hagenberg.at
[2] Johannes Kepler University, Department of Economics
Altenbergerstraße 69, 4040 Linz, Austria
stefan.fink@jku.at

Abstract. Macro-economic models describe the dynamics of economic quantities. The estimations and forecasts produced by such models play a substantial role for financial and political decisions. In this contribution we describe an approach based on genetic programming and symbolic regression to identify variable interactions in large datasets. In the proposed approach multiple symbolic regression runs are executed for each variable of the dataset to find potentially interesting models. The result is a variable interaction network that describes which variables are most relevant for the approximation of each variable of the dataset. This approach is applied to a macro-economic dataset with monthly observations of important economic indicators in order to identify potentially interesting dependencies of these indicators. The resulting interaction network of macro-economic indicators is briefly discussed and two of the identified models are presented in detail. The two models approximate the help wanted index and the CPI inflation in the US.

Keywords: Genetic programming, Finance, Econometrics.

1 Motivation

Macro-economic models describe the dynamics of economic quantities of countries or regions, as well as their interaction on international markets. Macro-economic variables that play a role in such models are for instance the unemployment rate, gross domestic product, current account figures and monetary aggregates. Macro-economic models can be used to estimate the current economic conditions and to forecast economic developments and trends. Therefore macro-economic models play a substantial role in financial and political decisions.

© Springer-Verlag Berlin Heidelberg 2011

It has been shown by Koza that genetic programming can be used for econometric modeling [6], [7]. He used a symbolic regression approach to rediscover the well-known exchange equation relating money supply, price level, gross national product and velocity of money in an economy, from observations of these variables.

Genetic programming is an evolutionary method imitating aspects of biological evolution to find a computer program that solves a given problem through gradual evolutionary changes starting from an initial population of random programs [7]. Symbolic regression is the application of genetic programming to find regression models represented as symbolic mathematical expressions. Symbolic regression is especially effective if little or no information is available about the studied system or process, because genetic programming is capable to evolve the necessary structure of the model in combination with the parameters of the model.

In this contribution we take up the idea of using symbolic regression to generate models describing macro-economic interactions based on observations of economic quantities. However, contrary to the constrained situation studied in [6], we use a more extensive dataset with observations of many different economic quantities, and aim to identify all potentially interesting economic interactions that can be derived from the observations in the dataset. In particular, we describe an approach using GP and symbolic regression to generate a high level overview of variable interactions that can be visualized as a graph.

Our approach is based on a large collection of diverse symbolic regression models for each variable of the dataset. In the symbolic regression runs the most relevant input variables to approximate each target variable are determined. This information is aggregated over all runs and condensed to a graph of variable interactions providing a coarse grained high level overview of variable interactions.

We have applied this approach on a dataset with monthly observations of economic quantities to identify (non-linear) interactions of macro-economic variables.

2 Modeling Approach

The main objective discussed in this contribution is the identification of all potentially interesting models describing variable relations in a dataset. This is a broader aim than usually followed in a regression approach. Typically, modeling concentrates on a specific variable of interest (target variable) for which an approximation model is sought. Our aim resembles the aim of data mining, where the variable of interest is often not known a-priori and instead all quantities are analyzed in order to find potentially interesting patterns [3].

2.1 Comprehensive Symbolic Regression

A straight forward way to find all potentially interesting models in a data set is to execute independent symbolic regression runs for all variables of the dataset building a large collection of symbolic regression models. This approach of comprehensive symbolic regression over the whole dataset is also followed in this contribution.

Especially in real world scenarios there are often dependencies between the observed variables. In symbolic regression the model structure is evolved freely, so any combination of input variables can be used to model a given target variable. Even if all input variables are independent, a given function can be expressed in multiple different ways which are all semantically identical. This fact makes the interpretation of symbolic regression models difficult as each run produces a structurally different result. If the input variables are not independent, for instance a variable x can be described by a combination of two other variables $yandz$, this problem is emphasized, because it is possible to express semantically equivalent functions using differing sets of input variables. A benefit of the comprehensive symbolic regression approach is that dependencies of all variables in the dataset are made explicit in form of separate regression models. When regression models for dependencies of input variables are known, it is possible to detect alternative representations.

Collecting models from multiple symbolic regression runs is simple, but it is difficult to detect the actually interesting models [3]. We do not discuss interestingness measures in this contribution. Instead, we propose a hierarchical approach for the analysis of results of multiple symbolic regression runs. On a high level, only aggregated information about relevant input variables for each target variable is visualized in form of a variable interaction network. If a specific variable interaction seems interesting, the models which represent the interaction can be analyzed in detail.

Information about relevant variable interactions is implicitly contained in the symbolic regression models and distributed over all models in the collection. In the next section we discuss variable relevance metrics for symbolic regression which can be used to determine the relevant input variables for the approximation of a target variable.

2.2 Variable Relevance Metrics for Symbolic Regression

Information about the set of input variables necessary to describe a given dependent variable is often valuable for domain experts. For linear regression modeling, powerful methods have been described to detect the relevant input variables through variable selection or shrinkage methods [4]. However, if non-linear models are necessary then variable selection is more difficult. It has been shown that genetic programming implicitly selects relevant variables [8] for symbolic regression. Thus, symbolic regression can be used to determine relevant input variables even in situations where non-linear models are necessary.

A number of different variable relevance metrics for symbolic regression have been proposed in the literature [12]. In this contribution a simple frequency-based variable relevance metric is proposed, that is based on the number of variable references in all solution candidates visited in a GP run.

2.3 Frequency-Based Variable Relevance Metric

The function $\text{relevance}_{\text{freq}}(x_i)$ is an indicator for the relative relevance of variable x_i. It is calculated as the average relative frequency of variable references $\text{freq}_{\%}(x_i, \text{Pop}_g)$ in population Pop_g at generation g over all G generations of one run,

$$\text{relevance}_{\text{freq}}(x_i) = \frac{1}{G}\sum_{g=1}^{G} \text{freq}_{\%}(x_i, \text{Pop}_g). \tag{1}$$

The relative frequency $\text{freq}_{\%}(x_i, \text{Pop})$ of variable x_i in a population is the number of references $\text{freq}(x_i, \text{Pop})$ of variable x_i over the number of all variable references,

$$\text{freq}_{\%}(x_i, \text{Pop}) = \frac{\sum_{s \in \text{Pop}} \text{RefCount}(x_i, s))}{\sum_{k=1}^{n} \sum_{s \in \text{Pop}} \text{RefCount}(x_k, s)}, \tag{2}$$

where the function $\text{RefCount}(x_i, s)$ simply counts all references to variable x_i in model s.

The advantage of calculating the variable relevance for the whole run instead of using only the last generation is that the dynamic behavior of variable relevance over the whole run is taken into account. The relevance of variables typically differs over multiple independent GP runs, because of the non-deterministic nature of the GP process. Therefore, the variable relevancies of one single GP run cannot be trusted fully as a specific variable might have a large relevance in a single run simply by chance. Thus, it is desirable to analyze variable relevance results over multiple GP runs in order to get statistically significant results.

3 Experiments

We applied the comprehensive symbolic regression approach, described in the previous sections, to identify macro-economic variable interactions. In the following sections the macro-economic dataset and the experiment setup are described.

3.1 Data Collection and Preparation

The dataset contains monthly observations of 33 economic variables and indexes from the United States of America, Germany and the Euro zone in the time span from 01/1980 – 07/2007 (331 observations). The time series were downloaded from various sources and aggregated into one large dataset without missing values.

Some of the time series in the dataset have a general rising trend and are thus also pairwise strongly correlated. The rising trend of these variables is not particularly interesting, so the derivatives (monthly changes) of the variables are studied instead of the absolute values. The derivative values ($d(x)$ in Figure 1) are calculated using the five point formula for the numerical approximation of the derivative [10] without prior smoothing.

Table 1. Genetic programming parameters

Parameter	Value
Population size	2000
Max. generations	150
Parent selection	Tournament (group size = 7)
Replacement	1-Elitism
Initialization	PTC2 [9]
Crossover	Sub-tree-swapping
Mutation	7% One-point, 7% sub-tree replacement
Tree constraints	Dynamic depth limit (initial limit = 7)
Model selection	Best on validation
Stopping criterion	$\rho(\text{Fitness}_{\text{train}}, \text{Fitness}_{\text{val}}) < 0.2$
Fitness function	R^2 (maximization)
Function set	+, -, *, /, avg, log, exp, sin
Terminal set	constants, variables, lagged variables (t-12) . . . (t-1)

3.2 Experiment Configuration

The goal of the modeling step is to identify the network of relevant variable interactions in the macro-economic dataset. Thus, several symbolic regression runs were executed to produce approximation models for each variable as a function of the remaining 32 variables in the dataset. In this step symbolic regression models are generated for each of the 33 variables in separate GP runs. For each target variable 30 independent runs are executed to generate a set of different models for each variable.

The same parameter settings were used for all runs. Only the target variable and the list of allowed input variables were adapted. The GP parameter settings for our experiments are specified in Table 1. We used rather standard GP configuration with tree-based solution encoding, tournament selection, subtree swapping crossover, and two mutation operators. The fitness function is the squared correlation coefficient of the model output and the actual values of target variables. Only the final model is linearly scaled to match the location and scale of the target variable [5]. The function set includes arithmetic operators (division is not protected) and additionally symbols for the arithmetic mean, the logarithm function, the exponential function and the sine function. The terminal set includes random constants and all 33 variables of the dataset except for the target variable. The variable can be either non-lagged or lagged up to 12 time steps. All variables contained in the dataset are listed in Figures 1 and 2.

Two recent adaptations of the algorithm are included to reduce bloat and overfitting. Dynamic depth limits [11] with an initial depth limit of seven are used to reduce the amount of bloat. An internal validation set is used to reduce the chance of overfitting. Each solution candidate is evaluated on the training and on the validation set. Selection is based solely on the fitness on the training set; the fitness on the validation set is used as an indicator for overfitting. Models which have a high training fitness but low validation fitness are likely to be overfit. Thus, the Spearman's rank correlation $\rho(\text{Fitness}_{\text{train}}, \text{Fitness}_{\text{val}})$ of training- and validation fitness of all solution candidates in the population is calculated after each generation. If the correlation of training- and validation fitness in the population drops below a certain threshold the algorithm is stopped.

The dataset has been split into two partitions; observations 1–300 are used for training, observations 300–331 are used as a test set. Only observations 13–200 are used for fitness evaluation, the remaining observations of the training set are used as internal validation set for overfitting detection and for the selection of the final (best on validation) model.

4 Results

For each variable of the dataset 30 independent GP runs have been executed using the open source software HeuristicLab. The result is a collection of 990 models, 30 symbolic regression models for each of the 33 variables generated in 990 GP runs. The collection of all models represents all identified (non-linear) interactions between all variables. Figure 1 shows the box-plot of the squared Pearson's correlation coefficient (R^2) of the model output and the original values of the target variable on the test set for the 30 models for each variable.

4.1 Variable Interaction Network

In Figure 2 the three most relevant input variables for each target variable are shown where an arrow ($a \rightarrow b$) means that variable a is a relevant variable for modeling variable b. In the interaction network variable a is connected to b ($a \rightarrow b$) if a is among the top three most relevant input variables averaged over all models for variable b, where the variable relevance is calculated using the metric shown in Equation 1. The top three most important input variables are determined for each of the 33 target variables in turn and GraphViz is used to layout the resulting network shown in Figure 2.

The network of relevant variables shows many strong double-linked variable relations. GP discovered strongly related variables, for instance *exports* and *imports* of Germany, *consumption* and *existing home sales*, *building permits* and *new home sales*, *Chicago PMI* and *non-farm payrolls* and a few more. GP also discovered a chain strongly related variables connecting the *producer price indexes* of the euro zone, Germany and the US with the *US CPI inflation.*

A large strongly connected cluster that contains the variables *unemployment*, *capacity utilization*, *help wanted index*, *consumer confidence*, *U.Mich. expectations*, *U.Mich. conditions*, *U.Mich. 1-year inflation*, *building permits*, *new home sales*, and *manufacturing payrolls* has also been identified by our approach.

Outside of the central cluster the variables *national activity index*, *CPI inflation*, *non-farm payrolls* and *leading indicators* also have a large number of outgoing connections indicating that these variables play an important role for the approximation of many other variables.

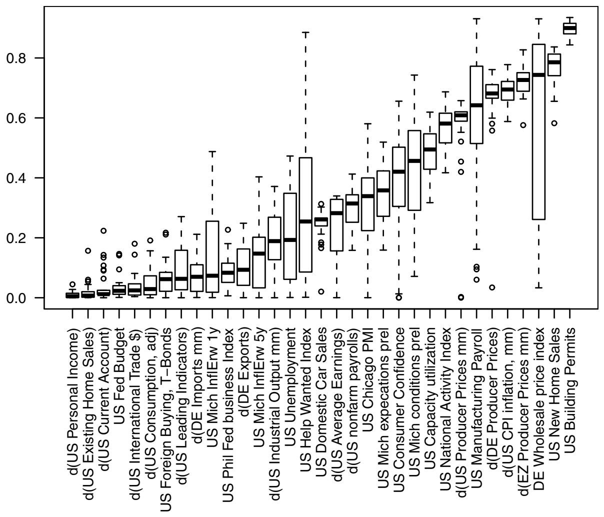


Fig. 1. Box-plot of the squared Pearson's correlation coefficient (R^2) of the model output and actual values on the test set for each variable

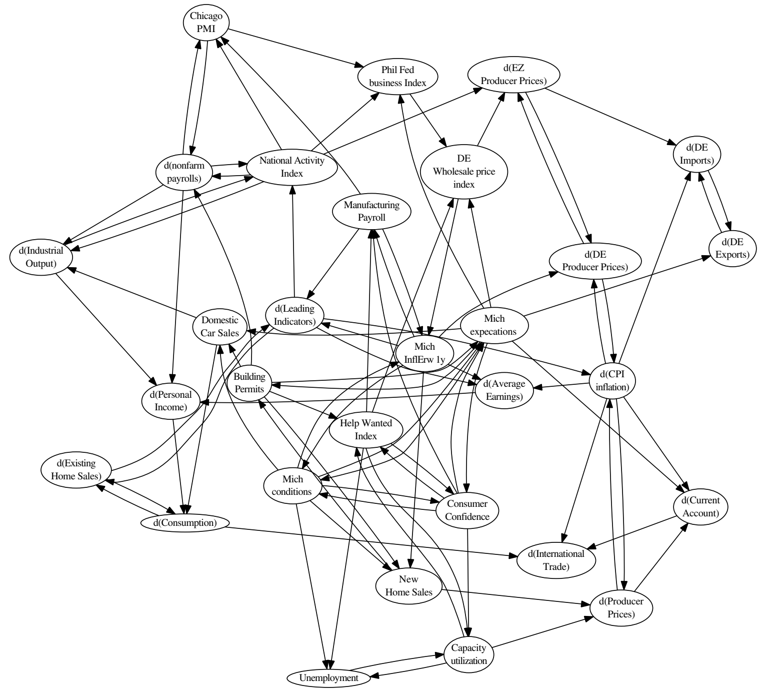


Fig. 2. Variable interaction network of macro-economic variables identified through comprehensive symbolic regression and frequency-based variable relevance metrics. This figure has been plotted using GraphViz.

4.2 Detailed Models

The variable interaction network only provides a course grained high level view on the identified macro-economic interactions. To obtain a better understanding of the identified macro-economic relations it is necessary to analyze single models in more detail. Because of space constraints we cannot give a full list of the best model identified for each variable in the data set. We selected two models for the *Help wanted index* and *CPI inflation* instead, which are discussed in more detail in the following sections.

The help wanted index is calculated from the number of job advertisements in major newspapers and is usually considered to be related to the unemployment rate [2], [1]. The model for the *help wanted index* shown in Equation 3 has a R^2 value of 0.82 on the test set. The model has been simplified manually and constant factors are not shown to improve comprehensibility. The model includes the *manufacturing payrolls* and the *capacity utilization* as relevant factors. Interestingly, the unemployment rate which was also available as input variable is not used, instead other indicators for economic conditions (*Chicago PMI*, *U. Mich cond.*) are included in the model. Interestingly the model also includes the *building permits* and *wholesale price index of Germany*.

$$\begin{aligned}\text{Help wanted index} = {} & \text{Building permits} + \text{Mfg payroll} + \text{Mfg Payroll}(t-5)\\ & + \text{Capacity utilization} + \text{Wholesale price index (GER)} \\ & + \text{Chicago PMI} + \text{U. Mich cond.}(t-3)\end{aligned} \tag{3}$$

Figure 3 shows a line chart for the actual values of the *help wanted index* in the US and the estimated values of the model (Equation 3) over the whole time span covered by the dataset.

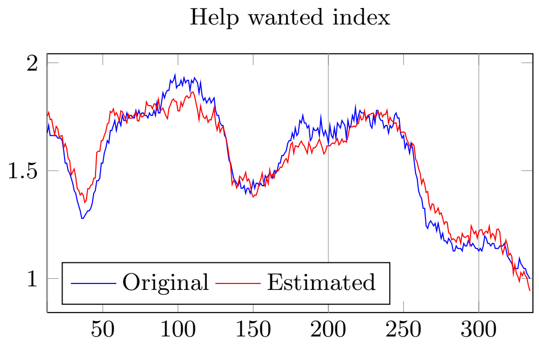


Fig. 3. Line chart of the actual value of the *US Help wanted index* and the estimated values produced by the model (Equation 3). Test set starts at index 300.

The *consumer price index* measures the change in prices paid by customers for a certain market basket containing goods and services, and is measure for the inflation in an economy. The output of the model for the *CPI inflation* in the US shown in Equation 4 is very accurate with a squared correlation coefficient of 0.93 on the test set. This model has also been simplified manually and again constant factors are not shown to improve comprehensibility. The model approximates the *consumer price index* based on the *unemployment*, *car sales*, *New home sales*, and the *consumer confidence*.

$$\begin{aligned}\text{CPI inflation} = {} & \text{Unemployment} + \text{Domestic car sales} + \text{New home sales} \\ & + \log(\text{New home sales}(t-4) + \text{New home sales}(t-2) \\ & + \text{Consumer conf.}(t-1) + \text{Unemployment}(t-5))\end{aligned} \tag{4}$$

Figure 4 shows a line chart for the actual values of the *CPI inflation* in the US and the estimated values of the model (Equation 4) over the whole time span covered by the dataset. Notably the drop of the CPI in the test set (starting at index 300) is estimated correctly by the model.

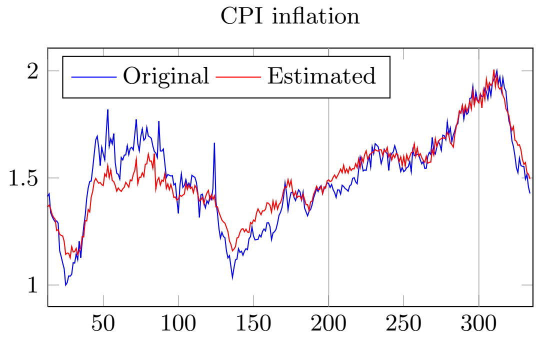


Fig. 4. Line chart of the actual value of the *US CPI inflation* and the estimated values produced by the model (Equation 4)

5 Conclusion

The application of the proposed approach on the macro-economic dataset resulted in a high level overview of macro-economic variable interactions. In the experiments we used dynamic depth limits to counteract bloat and an internal validation set to detect overfitting using the correlation of training- and validation fitness. Two models for the *US Help wanted index* and the *US CPI inflation* have been presented and discussed in detail. Both models are rather accurate also on the test set and are relatively comprehensible.

We suggest using this approach for the exploration of variable interactions in a dataset when approaching a complex modeling task. The visualization of variable interaction networks can be used to give a quick overview of the most relevant interactions in a dataset and can help to identify new unknown interactions. The variable interaction network provides information that is not apparent from analysis of single models, and thus supplements the information gained from detailed analysis of single models.

Acknowledgments. This work mainly reflects research work done within the Josef Ressel-center for heuristic optimization “Heureka!” at the Upper Austria University of Applied Sciences, Campus Hagenberg. The center “Heureka!” is supported by the Austrian Research Promotion Agency (FFG) on behalf of the Austrian Federal Ministry of Economy, Family and Youth (BMWFJ).

References

1. Abraham, K.G., Wachter, M.: Help-wanted advertising, job vacancies, and unemployment. Brookings Papers on Economic Activity, 207–248 (1987)
2. Cohen, M.S., Solow, R.M.: The behavior of help-wanted advertising. The Review of Economics and Statistics 49(1), 108–110 (1967)
3. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. The MIT Press, Cambridge (2001)
4. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning - Data Mining, Inference, and Prediction. Springer, Heidelberg (2009)
5. Keijzer, M.: Scaled symbolic regression. Genetic Programming and Evolvable Machines 5(3), 259–269 (2004)
6. Koza, J.R.: A genetic approach to econometric modeling. In: Sixth World Congress of the Econometric Society, Barcelona, Spain (1990)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
8. Langdon, W.B., Buxton, B.F.: Genetic programming for mining DNA chip data from cancer patients. Genetic Programming and Evolvable Machines 5(3), 251–257 (2004)
9. Luke, S.: Two fast tree-creation algorithms for genetic programming. IEEE Transactions on Evolutionary Computation 4(3), 274–283 (2000)
10. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++: The Art of Scientific Computing. Cambridge University Press, Cambridge (2002)
11. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines 10(2), 141–179 (2009)
12. Vladislavleva, K., Veeramachaneni, K., Burland, M., Parcon, J., O’Reilly, U.M.: Knowledge mining with genetic programming methods for variable selection in flavor design. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010), pp. 941–948 (2010)

Learning and Predicting Financial Time Series by Combining Natural Computation and Agent Simulation

Filippo Neri

Dept. of Computer and System Science, University of Naples,
via Claudio 21, 80125 Naples, Italy
filipponeri@yahoo.com

Abstract. We investigate how, by combining natural computation and agent based simulation, it is possible to model financial time series. The agent based simulation can be used to functionally reproduce the structure of a financial market while the natural computation technique finds the most suitable parameter for the simulator. Our experimentation on the DJIA time series shows the effectiveness of this approach in modeling financial data. Also we compare the predictions made by our system to those obtained by other approaches.

Keywords: Agent based modeling, natural computation, financial markets, prediction of the DJIA time series.

1 Introduction

The modeling of financial time series is a challenging research task for scientists working in several fields including Economics, Statistics, and Computer Science. In particular, computer scientists have recently developed novel research areas like Agent Based Modeling [2,6,15] and Agent-based Computational Economics [8,13,22] whose methodologies have found challenging applications in the study of financial markets. However, considering that financial markets remain substantially unpredictable, research contributions on their behavior are actively sought for. In this paper, we will focus our attention on a learning simulation system able to learn an agent based model (the simulation component) of a financial time series by exploiting a natural computation technique (the learning component).

Existing research on modeling financial markets in the artificial intelligence community usually falls into one of the following approaches:

a) evaluating or learning trading strategies for some given financial instruments (commodities, bonds, shares, derivatives, etc.), for instance [4,5,9,20];
b) building virtual markets in order to study phenomena like price bubble, as an instance [1,8,21];
c) and modeling the values/prices (and their evolution) for some financial assets, for instance [12,14].

The research reported in this paper belongs to the last category of investigation: here we will show how a Learning Financial Agent Based Simulator (L-FABS) can

© Springer-Verlag Berlin Heidelberg 2011

approximate the time series of the DJIA index under many experimental settings. We will also compare its performances with respect to other approaches.

The rest of the paper is organized as follows: Section 2 and 3 describe the component of our system. In Section 4, the experimental evaluation of the system is reported, while in Section 5 modeling approaches used by other researchers in modeling financial time series are compared to our system. Finally, in Section 6 the conclusions are drawn.

2 The Simple Financial Agent Based Simulator

Before describing the architecture of our system, we want to point out the two main assumptions that we have made about the economic behavior of an individual. We hypothesize that each investment decision for each individual depends on two main factors:

a) his/her propensity to take some risks today, by buying a financial asset, in exchange for a future uncertain reward, when perceiving the cash flows generated the asset: dividends or sale of the asset. To provide an example, this is the situation faced by a investor that has to decide if she/he prefers to invest in a Treasury Bond (very safe, but generating a low return), or in a corporate bond (less safe than a treasury bond, but generating an higher total return), or in stocks (even higher risk of losing the capital, if the company were to fail, but generating the highest return). We believe that the investment decision will be strongly influenced by the investor's risk/reward profile which we will model as a risk/reward propensity rate in the system.
b) the common and public consensus about the future behavior of the market. Public knowledge about the economic outlook diffused by financial news, economic reports, etc. will influence the investment decision of every one operating in the market. If the economic outlook is believed to be negative, on average people will tend to sell some of their riskier assets. If the economic outlook is believed to be positive, investors tend to buy some riskier assets. In the paper, we call market sentiment, or just sentiment, the common perception of the economy outlook and we will include it in our system.

We start now describing the basic component of our simulator: the FinancialAgent. The FinancialAgent implements the simple decision making process underlying the investment decision of buying, selling or holding an asset. The pseudo code for S-FABS is:

```
FinancialAgent(AgentId, Sentiment)
    Retrieve TotalAssets, InvestedAssets, and RiskRewardRate for AgentId
    BuyThreshold = RiskRewardRate × Sentiment
    SellThreshold = (1 - ((1 - BuyThreshold) / 2)
    Extract a random number Number in the interval (0,1)
    If Number < BuyThreshold Then // buy some assets
        InvestedAssets = InvestedAssets + 2% × TotalAssets
    If Number > SellThreshold Then // sell some assets
        InvestedAssets = InvestedAssets - 2% × TotalAssets
    Update TotalAssets, InvestedAssets for AgentId
```

Said in plain words, a Financial Agent will decide to buy some assets with probability P(X<BuyThreshold), it will hold to its assets with probability P(BuyThreshold<X<SellThreshold), and it will sell some of its assets with probability P(SellThreshold<X). The probability to sell some assets is half that of buy in order to account for the bias toward investing over selling that is shown by investors in real world. As it can be seen in the algorithm, the probabilities are also dependent on the current level of sentiment about the economy. In fact the probability to buy assets increases along with the sentiment, while the probability to sell assets decreases when the sentiment is raising and vice versa. In addition, the code will take care to retrieve and update the status of each financial agent that is here represented by the variables: TotalAssets, InvestedAssets, and RiskRewardRate.

Given the FinancialAgent pseudo code, the algorithm for simulating an individual, a simulation of the entire financial market can be obtained by creating several FinancialAgents, each one with its own status in terms of own assets, invested assets and risk/reward propensity, and then performing a sequence of investment rounds where each agent decides if buying, selling, or holding taking into account the current Sentiment value. At the end of each round, it is possible to measure the percentage of invested assets, this percentage can then be used as an estimated for one data point of the target time series. If the simulation is repeated for n rounds, the output will represent an estimate for an n-length time series. After explaining what the financial simulator S-FABS does, the algorithm follows. S-FABS take as input the vector of risk/rewards propensity rates for each Financial Agent. During each round, the risk/rewards propensity rates are used in combination with the current value of the economic sentiment by each Financial Agent. We will comment on the value of the Sentiment variable in the experimental section of the paper as it will be subject of learning under some of the experimental set ups. The same consideration hold for the risk/reward propensity rate for each type of investor.

```
S-FABS(the vector of RiskReward rates of the agents)
  Repeat for a given number of days
    Calculate the Sentiment for the current day
    For each agent AgentID do
      FinancialAgent(AgentID, Sentiment)
    Predicted Value for the current day = Total Invested Assets / Total Assets
```

A final point about the Financial Agents in S-FABS. In our study, we employ four types or classes of Financial Agents to capture the richness in investment decisions and in size of financial transactions that occur in real financial markets. The four types of investors we model are: individual investors (and the likes), banks (and the likes), hedge funds (and the likes), and central banks (and the likes). They differ in term of the size of the assets they can invest in financial markets and for their risk/reward appetite. In addition, their numerical presence is also different. Here are the values we used during our simulations:

Investor type	Total Assets (in millions)	Over 100 Investors
Individual	0.1	30
Funds	100	20
Banks	1000	49
Central Banks	10000	1

The figures in the table are to read only as a rough approximation for the average composition of the investors operating in the financial markets. Specific reasons for choosing four types of investors and for setting their parameters include: common view about who operates in the markets, the desire to keep the model simple while showing how it can preserve the diversity of the investors, and personal conversations with investment managers [3].

3 The Complete Learning Simulator: Combining Simulated Annealing with S-FABS

In this section, we describe how a learning capability can be added to S-FABS, by using Simulated Annealing, so that it may find the best model for a given time series. Learning in S-FABS consists of finding the vector of the parameters which control the simulation. In particular, learning consists of finding the vector of risk/reward propensity rates (plus any additional parameter object of experimentation) that approximates a given time series with a minimum error. This chosen learning framework allows for decoupling the learning phase from the simulation phase thus making possible to select as a learning techniques any of the many machine learning algorithms able to find a vector of values while minimizing a given error function. Examples of suitable machine learning algorithms include among the others: genetic algorithms [7,17], decision trees [18], neural networks [19], simulated annealing [11]. Given the possibility to select any of these learning methods, we decided to use Simulated Annealing because it emerges from the literature that evolutionary learning produces good results even when little or any domain knowledge is available. Because of page limit, we just say here that the exploited Simulated Annealing algorithm and its parameter settings are those described in [11].

For the error function to be minimized, we need to select one that can evaluate how well two time series are similar: the Mean Average Percentage Error (MAPE) is a suitable choice. The MAPE is commonly used in Statistics when two data samplings have to be compared in term of a non dimensional measure such as an percentage of the absolute difference between the data. The Mean Average Percentage Error is defined as:

$$MAPE(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{x_i - y_i}{x_i}\right|$$

Given two time series X and Y, the lower the MAPE value, the closer the two are. The MAPE value will provide a useful indication of how well a learning system has been able to approximate a target time series thus expressing the learning error of a found model.

4 Empirical Evaluation of L-FABS

In order to empirically evaluate L-FABS, we need to select some financial time series as datasets to work with. As usual with machine learning systems, we will train L-FABS on a part of the dataset, the learning set, and then we will use the remaining part of the dataset as test set to assess the performances of the learned model. The selected dataset consists of:

Dataset - learning set: DJIA close value data from 3 Jan 1994 to 17 Dec 2003 and test set: DJIA close value data from 18 Dec 2003 to 23 Oct 2006. The dataset has been freely acquired from the finance section of yahoo.com.

The reason for selecting this dataset is that it has been used to test other learning algorithms so the obtained approximation results can act as comparative measures for evaluating how our systems performed with respect to other approaches.

In the performed experiments, we will evaluate S-FABS when estimating the next value of the time series (the value of the next trading day) and the seven days ahead value of the time series. The seven days ahead prediction has been selected as it is the most far ahead prediction made by other learning systems and thus can serve as an interesting comparison data.

Before explaining the obtained results, the function for determining the market mood, the Sentiment, has to be defined. We want to stress here that our approach does not impose any restriction on how the market mood is determined. For our experiments, we implemented the Sentiment function as follows to keep it as simple and as general as possible:

```
function Sentiment(time, mavgDays)
    begin=time-1
    end=time-mavgDays
    if (MAVG(PredictedData, begin, end) <
        MAVG(RealData, begin, end))
        then return($\alpha$)
        else return($\beta$)
```

The MAVG function is defined as:

$$\text{MAVG(index,t,n)} = \sum_{k=0}^{n-1} index(t-k)/n$$

The variables RealData and PredictedData give access to the time series of the real and predicted values. The values α and β will assume the constant values of 0.65 and 0.30 (as empirically determined in an earlier work [16]) in a set of experiments, while will be determined by the learning algorithm itself in another set of experiments. According to our definition of the Sentiment, the outlook of the real market is considered bullish if the moving average (MAVG) of the predicted time series is lower than the moving average of the real data. If this is the case, the Sentiment value is set to an high value so that a bullish mood is communicated to the Financial Agents in S-FABS. The opposite happens if the predictions of the system have been higher than the real data. In our experiments will invoke the Sentiment function as either Sentiment(round,1), case identified in the following with S1, or Sentiment(round,5), case identified with S2. In case S1, only the previous day values of the two time series is used, while in case S2, the averages of the latest previous five days in the two time series are used to estimate the market mood.

We are now ready to discussed the experimental findings as reported in tables 1 and 2. The reported results are averaged over 10 runs of the same experimental setting and show the forecast errors are measured by MAPE on the test sets. In tables 1 and 2, we show the performances of L-FABS by allowing the learning algorithm to run for 200 and 400 rounds respectively. The columns in the tables stand for: "values for α and β" reports the values for the α and β parameters to be used in the Sentiment function as described above; "Sentiment" indicates if the Sentiment value is calculated with modality S1 or S5; "Day to predict" indicates the number of days ahead for which a prediction of the time series is made; and, finally, the measured errors on the test set are reported in terms of the MAPE. Also the table is divided in two parts: the first four rows are experiments with constant values for the α and β parameters, while the last four rows represents experiments where the α and β are learned by the simulated annealing together with the risk/reward propensity rates for each investor type. This means that the experimental set up in row one is similar to the experimental setting of row five with the only difference that the learning task given to the learning algorithm (the simulated annealing) in row one is to find only the vector of risk/reward propensity rates whereas in row five the objective of learning is to find both the vector of risk/reward propensity rates plus the value for the α and β parameters. From the experimental findings it appears that the predictions of close values for the next day of the DJIA are more accurate than the predictions made for the seven days ahead values. This finding confirms the intuitive experience that the farther a prediction is moved into the future, the less accurate it will be.

Table 1. Experimental results on dataset DJIA using 200 rounds of SA

values for α and β	Sentiment	Day to predict	MAPE %
0.65, 0.30	S1	1	0.76
0.65, 0.30	S5	1	0.74
0.65, 0.30	S1	7	1.48
0.65, 0.30	S5	7	1.51
0.40, 0.31	S1	1	0.62
0.31, 0.29	S5	1	0.67
0.51, 0.47	S1	7	1.35
0.43, 0.44	S5	7	1.52

Table 2. Experimental results on dataset DJIA using 400 rounds of SA

values for α and β	Sentiment	Day to predict	MAPE %
0.65, 0.30	S1	1	0.74
0.65, 0.30	S5	1	0.69
0.65, 0.30	S1	7	1.48
0.65, 0.30	S5	7	1.47
0.55, 0.48	S1	1	0.57
0.51, 0.48	S5	1	0.58
0.50, 0.43	S1	7	1.39
0.53, 0.43	S5	7	1.45

Also it emerges that using only the previous day close for estimating the market mood, Sentiment S1, is as good as using the moving average of the latest five days close values for the index, case S5, in making a good prediction. Comparing the results in the first four rows with the latter four rows in the table, it is also evident that when the system is let learning the parameters α and β, which specify how large is the Sentiment value in case of positive or negative feelings, L-FABS is able to produce slightly better forecasts.

It is also important to note that the MAPE errors across the several experimental settings tend to stay very close for similar experimental setup, compare experiments across the tables 1 and 2. This is an important feature in a learning systems and it is called *robustness* that is the ability to display a consistent good behavior across a range of different parameter settings.

5 Experimental Comparison of L-FABS to Other Systems

For providing an exhaustive evaluation of L-FBAS, we will compare its performances with respect to those obtained on the same dataset by alternative approaches for which enough implementation details have been given in the literature [14] so they can act as an useful benchmark. In table 3, we compare the prediction errors, as given by the MAPE measure, for the DJIA time series, corresponding to the Dataset DJIA of the previous section, by using a Particle Swarm Optimization algorithm (PSO) [10] and a Multi-Layer Perceptron (MLP) [23], whose parameters have been set up as in [14], with our agent based approach L-FABS. The results for L-FABS are relative to a Sentiment determined as in case S1 and using 400 rounds of simulated annealing.

Just at a first glance, the results in table 3, shows that the forecasting errors of L-FABS are better than those obtained by of PSO and MLP. Moreover, it is evident, as observed in the previous section, that the forecasting error increases when farther into the future the prediction is to be made. And this holds for all the systems. This observation can be interpreted as evidence that the information contained in the time series up to the current time has a decreasing usefulness in predicting future values the farther we move ahead in time. This finding also confirms what we expect to happen in real world financial markets.

Table 3. Experimental results averaged on 10 runs for time series DJIA

Day to predict	PSO MAPE %	MLP MAPE %	L-FABS MAPE %
1	0.65	1.06	0.57
7	1.47	5.64	1.39

6 Conclusions

In the paper, we have described how an agent based modeling techniques combined with natural computation, simulated annealing, could result in a learning simulative system, L-FABS, able to find the model for a given financial time series. The agent based simulation can be used to functionally reproduce the structure of a financial market while the natural computation technique finds the most suitable parameter for the simulation. We have empirically evaluated the system, under several parameter settings, on the DJIA time series and the predictive power of the learning simulator has also been compared with respect to existing approaches. The main result of our paper is that: a conceptually simple learning simulator, combining agent based modeling and simulated annealing, can closely approximate the DJIA time series and provide forecasts for it comparable to those obtained by more sophisticated approaches.

References

1. Arthur, W.B., Holland, J.H., LeBaron, B., Palmer, R., Taylorm, P.: Asset pricing under endogenous expectation in an artificial stock market. In: The Economy as an Evolving Complex System II. Santa Fe Institute Studies in the Sciences of Complexity Lecture Notes, pp. 15–44 (1997)
2. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences 99(3), 7280–7287 (2002)
3. Cesa, A.: Discussion about how financial markets work: an investment manager perspective. Personal correspondance with the author (2009)
4. Creamer, G., Freund, Y.: Automated trading with boosting and expert weighting. Quantitative Finance 4(10), 401–420 (2010)
5. 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)
6. Epstein, J.M., Axtell, R.: Growing artificial societies: social science from the bottom up. The Brookings Institution, Washington, DC, USA (1996)
7. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
8. Hoffmann, A.O.I., Delre, S.A., von Eije, J.H., Jager, W.: Artificial multi-agent stock markets: Simple strategies, complex outcomes. In: Advances in Artificial Economics. LNEMS, vol. 584, pp. 167–176. Springer, Heidelberg (2006)
9. Kendall, G., Su, Y.: A multi-agent based simulated stock market - testing on different types of stocks. In: Congress on Evolutionary Computation, CEC 2003, pp. 2298–2305 (2003)

10. Kennedy, J., Eberhard, R.: Particle swarm optimization. In: Int. Conf. on Neural Networks, pp. 1942–1948. IEEE press, Los Alamitos (1995)
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
12. Kitov, I.: Predicting conocophillips and exxon mobil stock price. Journal of Applied Research in Finance 2, 129–134 (2009)
13. Lebaron, B.: Agent based computational finance: Suggested readings and early research. Journal of Economic Dynamics and Control 24, 679–702 (1998)
14. Majhi, R., Sahoo, G., Panda, A., Choubey, A.: Prediction of sp500 and djia stock indices using particle swarm optimization techniques. In: Congress on Evolutionary Computation 2008, pp. 1276–1282. IEEE Press, Los Alamitos (2008)
15. Neri, F.: Empirical investigation of word-of-mouth phenomena in markets: a software agent approach. WSEAS Transaction on Computers 4(8), 987–994 (2005)
16. Neri, F.: Using software agents to simulate how investors' greed and fear emotions explain the behavior of a financial market. In: WSEAS Conference ICOSSE 2009, Genoa, Italy, pp. 241–245 (2009)
17. Neri, F., Saitta, L.: Exploring the power of genetic search in learning symbolic classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-18(11), 1135–1142 (1996)
18. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, California (1993)
19. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. foundations, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
20. Schulenburg, S., Ross, P., Bridge, S.: An adaptive agent based economic model. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 1999. LNCS (LNAI), vol. 1813, pp. 263–284. Springer, Heidelberg (2000)
21. Takahashi, H., Terano, T.: Analyzing the influence of overconfident investors on financial markets through agent-based model. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (eds.) IDEAL 2007. LNCS, vol. 4881, pp. 1042–1052. Springer, Heidelberg (2007)
22. Tesfatsion, L.: Agent-based computational economics: Growing economies from the bottom up. Artif. Life 8(1), 55–82 (2002)
23. Zirilli, J.: Financial prediction using Neural Networks. International Thompson Computer Press (1997)

A Preliminary Investigation of Overfitting in Evolutionary Driven Model Induction: Implications for Financial Modelling

Clíodhna Tuite, Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon

Financial Mathematics and Computation Cluster
Natural Computing Research and Applications Group
Complex and Adaptive Systems Laboratory
University College Dublin, Ireland
cliodhna.tuite@gmail.com,
{alexandros.agapitos,m.oneill,anthony.brabazon}@ucd.ie

Abstract. This paper investigates the effects of early stopping as a method to counteract overfitting in evolutionary data modelling using Genetic Programming. Early stopping has been proposed as a method to avoid model overtraining, which has been shown to lead to a significant degradation of out-of-sample performance. If we assume some sort of performance metric maximisation, the most widely used early training stopping criterion is the moment within the learning process that an unbiased estimate of the performance of the model begins to decrease after a strictly monotonic increase through the earlier learning iterations. We are conducting an initial investigation on the effects of early stopping in the performance of Genetic Programming in symbolic regression and financial modelling. Empirical results suggest that early stopping using the above criterion increases the extrapolation abilities of symbolic regression models, but is by no means the optimal training-stopping criterion in the case of a real-world financial dataset.

1 Introduction

Overfitting is a commonly studied problem which arises in machine learning techniques such as Genetic Programming. A model is described as overfitting the training data if, despite having a high fit on the training examples, there exists another model which has better fitness on the data as a whole, despite not fitting the training data as well [9]. There are different reasons why overfitting can occur. The existence of noise in training samples can cause a model to be fit to the data which is more complex than the true underlying model [14]. For symbolic regression, an example would be fitting a high order polynomial to noisy data, which happens to pass through all training points, when the true function is in fact a lower order polynomial. Another cause of overfitting is bias in the training data. Overfitting is more likely to occur when the training sample size is small. The more data available to train on, the more likely we are to discover the true underlying model, and the less likely we are to settle on a spurious result.

© Springer-Verlag Berlin Heidelberg 2011

Overfitting is also more likely to occur in the presence of complexity. Complex models (for example symbolic regressions of multiple explanatory variables) are more likely to induce overfitting. Learning algorithms that are run for a long time are also more likely to trigger overfitting, than if they had been run for a shorter time period [1].

This paper aims to begin to explore the issue of overfitting in Grammatical Evolution [12,6] (a form of grammar-based Genetic Programming [8]), and provides case studies of overfitting in three symbolic regression problems, and a financial modelling example drawn from an instance of credit risk classification.

2 Model Induction

The underlying data generating process is unknown in many real-world financial applications. Hence, the task is often to deduce or "recover" an underlying model from the data. This usually isn't an easy task since both the model structure and associated parameters must be uncovered. Most theoretical financial asset pricing models make strong assumptions which are often not satisfied in real-world asset markets. They are therefore good candidates for the application of model induction tools, such as Grammatical Evolution, which are used to recover the underlying data generating processes [3].

Of course to use a model induction method effectively, that is, to ensure that the evolved models generalise beyond the training dataset, we must pay attention to overfitting. This study aims to highlight this important open issue in the field of Genetic Programming [13] and its implications for financial modelling.

3 Background

3.1 Model Generalisation

A crucial aspect of data-driven modelling is related to model generalisation, and many financial applications of evolutionary methods do not apply techniques to minimise overfitting. Model generalisation concerns the ability of the induced model to correctly represent the underlying structure of the data so as to make accurate predictions when presented with new data from the problem domain. Unfortunately, data-mining methodologies that iteratively refine a model on a set of training instances, as is the case of evolutionary methods, inherently suffer from model overfitting; they produce solutions with poor generalisation abilities. There has been a large amount of statistics and sibling machine learning methodologies to counteract the phenomenon of overfitting and produce models with competent out-of-sample performance.

3.2 Model Overtraining Avoidance through Early Stopping

A well-exercised technique for promoting the generalisation of an induced model is the procedure of *early training stopping* [16,9,7]. For most learning algorithms the training error decreases monotonically during training. If an independent validation dataset is used to measure the model's accuracy on unseen data, the validation error tends also to decrease in step with the training error as the model gradually approximates the underlying function. However, it is very often the case that the training data contain spurious and misleading regularities due to sampling. In the later stages of the training process, the model begins to exploit these idiosyncrasies in the training data and the validation error tends to increase again while the training error continues to decrease. This example of overfitting is described in [5]. One approach to avoid overfitting is to use the independent validation dataset as part of a heuristic that dictates the halting of the training process at the first minimum of the validation error. Under such a regime, the learner is trained using the training instances, however, in each learning iteration it is evaluated for both training and validation accuracy. Typically, the error on the validation set decreases along with the training error, but then tends to increase, an indication that the model may be overfitting the training instances, suggesting that the training phase should be stopped. It has been shown that halting the training phase before a minimum of the training error has been reached, represents a way of limiting the complexity of the induced model [2].

3.3 Grammatical Evolution: A Brief Introduction

In Grammatical Evolution [12,6], the process of evolution first involves the generation of a population of randomly generated binary (or integer) strings, the genotype. In the case of binary genomes, each set of B bits (where traditionally B=8) is converted into its equivalent integer representation. These integer strings are then mapped to a phenotype, or high-level program or solution, using a grammar, which encompasses domain knowledge about the nature of the solution. Therefore, a GE genome effectively contains the instructions of how to build a sentence in the language specified by the input grammar. Grammatical evolution is a form of what is known as grammar-based Genetic Programming [8], and has been applied to a broad range of problems, including many successful examples in financial modelling [4].

The grammar used in the experiments we performed can be found in Fig. 1. The grammar is composed of non-terminal and terminal symbols. Terminals (for example arithmetic operators) appear in the solution, whereas non-terminals can be further expanded into terminals and non-terminals. Here we can see that knowledge of the solution (that it will be constructed from arithmetic operators, mathematical functions, variables and constants) is encoded in the grammar. The mapping process involves the use of an integer from the genotype to choose a rule from the production rule currently being mapped. This process proceeds as follows. The first integer from the genotype is divided by the number of rules in the start symbol (`<expr>` in our example). The remainder from this division is used to select a rule from the grammar (for example, if the first integer was 8, the result of dividing 8 by the number of choices available for the `<expr>` production rule, which is 5, would result in the choice of the third rule - which is `<pre-op>(<expr>)`. The next integer in the genotype would then be used in the same way to map between `<pre-op>` and one of its constituent rules, and the third integer in the genotype would be used to map between `<expr>` and one of its constituent rules. This process continues until either all integers in the genotype have been used up, or our mapping process has resulted in the production of a phenotype (that is a structure comprised of only terminal symbols) [12].

```
<prog> ::= <expr>
<expr> ::= <expr> <op> <expr>  | ( <expr> <op> <expr> ) |
           <pre-op> ( <expr> ) | <protected-op> | <var>
<op> ::= + | * | -
<protected-op> ::= div( <expr>, <expr>)
<pre-op> ::= sin | cos | exp | inv | log
<var> ::= X | 1.0
```

Fig. 1. Grammar used in Symbolic Regressions

4 Experimental Setup

4.1 Symbolic Regression

Grammatical Evolution was used to fit models to 3 symbolic regression problems. Equations 1 through 3 show the target functions. The training dataset was comprised of 10 randomly generated points. The test dataset (which was not used to train the model) was comprised of 20 randomly generated points, 10 of which were drawn from the same range as the training data (to test how well the model interpolates, and to serve as a proxy for a validation dataset, see Section 5.1), and 10 of which were drawn from outside the range from which the training data were drawn (to test how well the model extrapolates).

$$Y = 0.6X^3 + 5X^2 - 10X - 25 \tag{1}$$

Training dataset range: [-5, 5]. Test dataset ranges: [-10, 10].

$$Y = 0.3X \times \sin 2X \tag{2}$$

Training dataset range: [-1, 1]. Test dataset ranges: [-2, 2].

$$Y = \exp X - 2X \tag{3}$$

Training dataset range: [-2, 2]. Test dataset ranges: [-4, 4].

These functions and ranges were chosen so that the target function would be trained using a biased sample. The bias resulted from training in a range in which the target function closely resembled an alternative function. Over a wider range than that from which the training data was drawn, the target function looked dramatically different from this alternative (for example, function 2 looked very like a quadratic in the training range (see Fig. 7), but as can be seen, it is truly a sine function). In this way, we engineered a situation in which overfitting was likely to take place. In each case, Grammatical Evolution was run on a population size of 100 individuals, for 50 generations, using Grammatical Evolution in Java [11]. The grammar used is shown in Fig. 1.

Fitness was evaluated by computing the mean squared error of the training points when evaluated on each individual (therefore the lower the fitness value, the better the evolved function fitted the training data).

$$MSE = \frac{\sum_{i=1}^{n} |targetY - phenotypeY|^2}{n} \tag{4}$$

4.2 The Case of a Financial Dataset

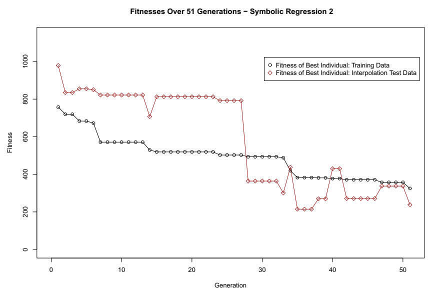
We also test the early stopping approach to model overfitting on a real world financial dataset from the UCI Machine Learning repository [10]. The financial dataset represents a binary classification problem of categorising credit card applications between those which are approved or rejected. The dataset contains 690 number of instances, and each instance has 15 attributes.

We employed GE to evolve non-linear discriminant functions that use the threshold value of zero to differentiate among the classes. The context-free grammar is represented below.

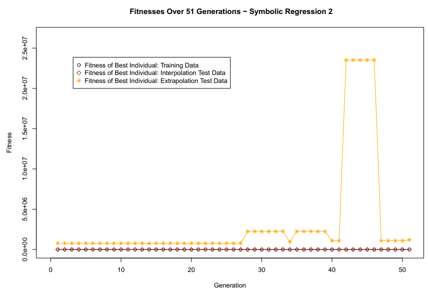
```
<prog> ::= <expr>
<expr> ::= <expr> <op> <expr> | <var>
<op> ::= + | * | - | /
<var> ::= instance attributes
```

Fig. 2. Grammar used in the financial credit classification problem

The GP algorithm employs a panmictic, generational, elitist genetic algorithm. The algorithm uses tournament selection with a tournament size of 7. The population size is set to 500 individuals, and the number of generations to 100. Ramped-half-and-half tree creation with a maximum depth of 6 is used to perform a random sampling of DTs during run initialisation. Throughout evolution, expression-trees are allowed to grow up to depth of 15. The evolutionary search combines standard subtree crossover with subtree mutation; a probability governing the application of each, set to 0.6 in favour of subtree crossover. We used the classification accuracy (CA) as the fitness function, but in order to convert it to a minimisation problem we assigned fitness using $1.0 - CA$. We split the original dataset into two random equally-sized subsamples with equal distribution of classes, serving as the training and validation (out-of-sample) datasets.

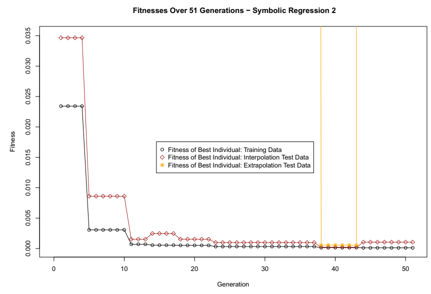


(a)

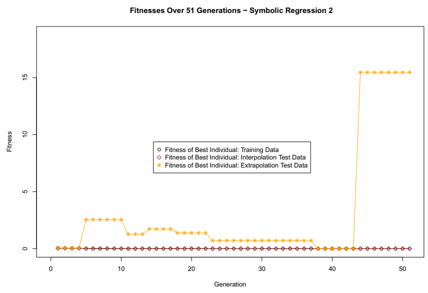


(b)

Fig. 3. Target Function 1



(a)



(b)

Fig. 4. Target Function 2, Example 1

5 Results and Discussion

5.1 Symbolic Regression

Figs. 3(a) through 6(b) are plots of the fitness of the best individual at each generation as evaluated on the training data, against the fitness of the best individual at each generation as evaluated on the test datasets, for four illustrative runs - one run each of target functions 1 and 3, and two runs of target function 2. Table 1 contains details on the fitness as evaluated on the test dataset, for 9 runs. It shows that stopping evolution before the specified number of generations had elapsed, would have led to the model extrapolating better beyond the range in which it was trained.

Early stopping has been described in Section 3.2. The validation dataset is not used to train the model, but instead is used to test the fitness of the model every once in a while (for example each generation, or at five generation intervals). If the fitness of the best individual as evaluated on the validation dataset disimproves, this is taken as an indication that the evolved model is overfitting the data, and evolution is stopped. (Test data is used as before to evaluate the fitness of the evolved model on out-of-sample data, after evolution has terminated, either prematurely (if early stopping has been deemed necessary), or after the specified number of generations has elapsed.)

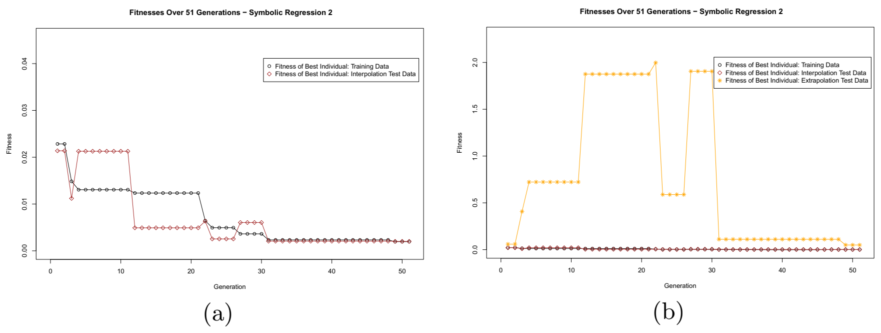


Fig. 5. Target Function 2, Example 2

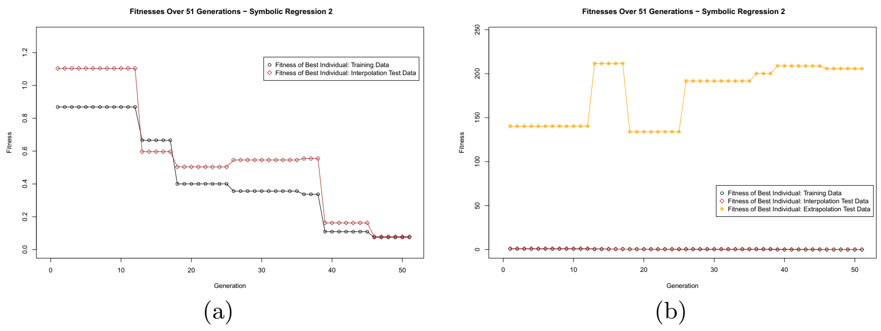


Fig. 6. Target Function 3

Table 1. Interpolation, Extrapolation fitnesses - Generations at which deteriorations took place

Target Function Number	1	1	1
Generation Interpolation Fitness First Disimproved (GIFFD)	4	10	4
Interpolation Fitness Better Generation GIFFD-1, or End of Run?	End of Run	End of Run	GIFFD-1
Extrapolation Fitness Better Generation GIFFD-1, or End of Run?	GIFFD-1	GIFFD-1	GIFFD-1
Best stopping point (Generation(s) of Lowest Extrapolation Fitness)?	14	7 - GIFFD-1	12
Target Function Number	2	2	2
Generation Interpolation Fitness First Disimproved (GIFFD)	14	3	4
Interpolation Fitness Better Generation GIFFD-1, or End of Run?	End of Run	End of Run	End of Run
Extrapolation Fitness Better Generation GIFFD-1, or End of Run?	GIFFD-1	End of Run	GIFFD-1
Best stopping point (Generation(s) of Lowest Extrapolation Fitness)?	38 - 43	49 - End of Run	5 - 8
Target Function Number	3	3	3
Generation Interpolation Fitness First Disimproved (GIFFD)	26	7	19
Interpolation Fitness Better Generation GIFFD-1, or End of Run?	End of Run	End of Run	GIFFD-1
Extrapolation Fitness Better Generation GIFFD-1, or End of Run?	GIFFD-1	GIFFD-1	GIFFD-1
Best stopping point (Generation(s) of Lowest Extrapolation Fitness)?	18 - GIFFD-1	7 - 10	8 - GIFFD-1

Since we explicitly chose target functions and ranges with an inherent bias, these symbolic regressions triggered overfitting, as expected. Table 1 shows the generation at which the fitness, as evaluated on the part of the test dataset used to measure the ability of the evolved model to interpolate in the training range (henceforth referred to as interpolation fitness), first disimproved. In 8 of the 9 runs described, the fitness as evaluated on the part of the test dataset used to measure the ability of the evolved model to extrapolate beyond the training

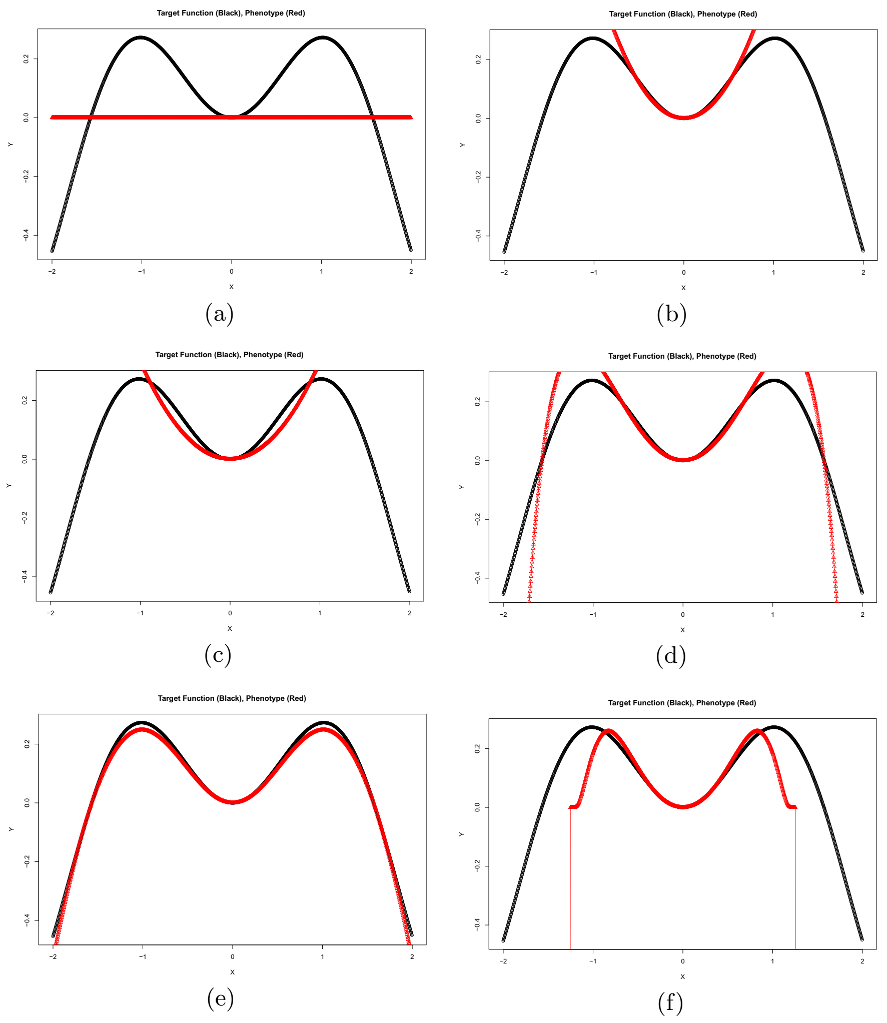


Fig. 7. (a) Generation 1 (b) Generation 5 (c) Generation 11 (d) Generation 23 (e) Generation 38 (f) Generation 44

range (henceforth referred to as extrapolation fitness), was better the generation immediately before the interpolation fitness first disimproved, than at the end of the run. Had we stopped evolution at this point, we would have produced a model that extrapolated better beyond the training range, than the model produced at the end of the run.

The data points from the test dataset drawn from the same range as the training dataset (and used to measure how well the evolved model is interpolating within the training range), can also be used as a proxy for a validation dataset.

[15] show that when training artificial neural networks, the first time the error on the validation set increases is not necessarily the best time to stop training, as the error on the validation set may increase and decrease after this first disimprovement. Such a pattern seems to exist in the runs we performed. In 5 of the 8 runs where early stopping would have made sense, the optimal generation at which to stop (the generation with the lowest extrapolation fitness value) came later than the generation at which the interpolation fitness first disimproved.

To give further insight into the evolutionary process that underlie the changes in fitness observed for the training and test data sets, the phenotype was plotted against the target function in the extrapolation range, at each generation. Fig. 7 shows a selection of these generational graphs for the first run of function 2.

Comparing Figs. 7 and 4(b), we can clearly see the correspondences between changes in the graphed phenotype over the generations, and changes in the fitness as evaluated on the extrapolation test data. Between generations 1 and 22, the extrapolation test fitness is either disimproving, or not improving by much. At generation 23 fitness improves significantly, and at generation 38, an extremely fit individual has been evolved, both with respect to the training and test set. The model extrapolates well. However, come generation 44, a much less fit function has been evolved. It's fitness on the training data has improved, but it's fitness on the extrapolation test data has drastically disimproved. If we look back at Fig. 4(b), we can clearly see both an extremely low value in the fitness on the extrapolation test data at generation 38, and an explosion in the value of the fitness on the extrapolation test data at generation 44.

5.2 Financial Dataset

We performed 100 independent evolutionary runs. Table 2 presents average performances of the best-of-generation individuals, on both training and validation sets, throughout the evolutionary process. Results suggest that there is no particular evidence of model overfitting; the validation performance curve monotonically decreases up until generation 80, at which point a slight degree of overtraining becomes apparent. This is evidenced by the average percentage change in the validation performance, which reaches a negative number between generations 80 and 90 (Table 3). The model performance in the case of early stopping at the generation that the validation error becomes a local minimum

Table 2. Training and Test Learning Curves for the classification problem. Averages of 100 evolutionary runs. Standard deviation in parentheses.

	Gen. 10	Gen. 20	Gen. 30	Gen. 40	Gen. 50
Training performance	0.25 (0.01)	0.24 (0.01)	0.23 (0.01)	0.22 (0.01)	0.22 (0.01)
Validation performance	0.29 (0.02)	0.29 (0.02)	0.28 (0.02)	0.28 (0.02)	0.28 (0.02)
	Gen. 60	Gen. 70	Gen. 80	Gen. 90	Gen. 100
Training performance	0.21 (0.01)	0.21 (0.01)	0.21 (0.01)	0.20 (0.01)	0.20 (0.01)
Validation performance	0.28 (0.02)	0.27 (0.02)	0.27 (0.02)	0.28 (0.02)	0.28 (0.02)

Table 3. Percentage change in Training and Testing performance. Averages of 100 evolutionary runs. Standard deviation in parentheses.

	Gen. 10-20	Gen. 20-30	Gen. 30-40	Gen. 40-50	Gen. 50-60
Training performance change (%)	5.2% (0.03)	3.7% (0.03)	2.7% (0.02)	1.7% (0.02)	1.7% (0.02)
Validation performance change (%)	2.0% (0.07)	1.9% (0.07)	0.7% (0.05)	0.6% (0.04)	0.2% (0.03)
	Gen. 60-70	Gen. 70-80	Gen. 80-90	Gen. 90-100	
Training performance change (%)	1.4% (0.01)	1.2% (0.01)	1.3% (0.01)	0.9% (0.01)	
Validation performance change (%)	0.2% (0.03)	0.3% (0.03)	-0.6% (0.03)	0.6% (0.03)	

Table 4. Performance statistics during early stopping. Averages of 100 evolutionary runs. Standard deviation in parentheses.

Early Stopping Generation	5.88 (6.07)
Early stopping training performance	0.27 (0.02)
Early stopping validation performance	0.31 (0.03)

for the first time, is summarised in Table 4. It is apparent that stopping at approximately generation 6 results in a model with validation performance of 0.31, which is clearly not the optimal point at which to stop, given that the best validation performance is 0.27, attained by generation 80 (Table 2).

Overall, this empirical result suggests that early training stopping at the first point where the validation error reaches a local minimum (assuming fitness minimisation) is by no means a reliable indication of overtraining. Future research needs to address the issue of early stopping with more sophisticated stopping criteria.

6 Conclusions and Future Work

In this study we set out to highlight a significant open issue in the field of Genetic Programming, namely generalisation of evolved solutions to unseen data, which has real world implications for all model induction methods, and can have serious financial implications when considered in the domain of financial modelling. Empirical investigations on four benchmark problems are undertaken. Three of the problems were drawn from the popular Genetic Programming domain of symbolic regression and the fourth problem was an instance of credit classification.

In summary the results illustrate the important role which the detection of overfitting during training can play, in order to improve the generalisation of the evolved models. What is also clear from these results is that further lessons need to be drawn from the machine learning literature on effective early stopping strategies, and the myriad of other strategies which have been adopted to avoid overfitting. The results on both classes of problem domain investigated here demonstrate that early stopping could be an effective strategy to improve generalisation, however, following a naive early stopping heuristic can lead to stopping *too early*.

Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant No. 08/SRC/FMC1389.

References

1. Becker, L.A., Seshadri, M.: Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Worcester Polytechnic Institute, Computer Science Technical Report (2003)
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1996)
3. Brabazon, A., Dang, J., Dempsey, I., O'Neill, M., Edelman, D.: Natural computing in finance: a review (2010)
4. Brabazon, A., O'Neill, M.: Biologically inspired algorithms for financial modelling. Springer, Heidelberg (2006)
5. Chauvin, Y.: Generalisation performance of overtrained back-propagation networks. In: EUROSIP Workshop, pp. 46–55 (1990)
6. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. Springer, Heidelberg (2009)
7. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. John Wiley and Sons, Chichester (2001)
8. Mckay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based Genetic Programming: a survey. Genetic Programming and Evolvable Machines 11(3-4), 365–396 (2010)
9. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
10. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)
11. O'Neill, M., Hemberg, E., Gilligan, C., Bartley, E., McDermott, J., Brabazon, A.: GEVA: grammatical evolution in Java. ACM SIGEVOlution 3(2), 17–22 (2008)
12. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary automatic programming in an arbitrary language. Springer, Netherlands (2003)
13. O'Neill, M., Vanneschi, L., Gustafson, S., Banzhaf, W.: Open issues in genetic programming. Genetic Programming and Evolvable Machines 11(3-4), 339–363 (2010)
14. Paris, G., Robilliard, D., Fonlupt, C.: Exploring overfitting in genetic programming. In: Artificial Evolution, pp. 267–277. Springer, Heidelberg (2004)
15. Prechelt, L.: Early stopping-but when? In: Neural Networks: Tricks of the trade, pp. 553–553 (1998)
16. Sarle, W.S.: Stopped training and other remedies for overfitting. In: Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, pp. 352–360 (1995)

On the Performance and Convergence Properties of Hybrid Intelligent Schemes: Application on Portfolio Optimization Domain

Vassilios Vassiliadis, Nikolaos Thomaidis, and George Dounias

Abstract. Hybrid intelligent algorithms, especially those who combine nature-inspired techniques, are well known for their searching abilities in complex problem domains and their performance. One of their main characteristic is that they manage to escape getting trapped in local optima. In this study, two hybrid intelligent schemes are compared both in terms of performance and convergence ability in a complex financial problem. Particularly, both algorithms use a type of genetic algorithm for asset selection and they differ on the technique applied for weight optimization: the first hybrid uses a numerical function optimization method, while the second one uses a continuous ant colony optimization algorithm. Results indicate that there is great potential in combining characteristics of nature-inspired algorithms in order to solve NP-hard optimization problems.

Keywords: Genetic Algorithm, Continuous ACO, Portfolio Optimization.

1 Introduction

Nowadays, complex constrained problems in various domains pose a great challenge both in academics and decision making. Traditional methodologies, from statistics and mathematics, fail to deal properly with these problems, and in most cases they converge to local optima. In recent years, there has been an ongoing trend in applying metaheuristic methodologies from the field of artificial intelligence due to the unique characteristics which enable them to tackle hard-solving problems.

Hybrid algorithms comprise two or more intelligent metaheuristic techniques. Their main advantage lies on the combination of characteristics of various intelligent schemes. Hybrid algorithms are well recognized for their searching strategies in complex solution domains as well as their ability in avoiding getting stuck in local optima.

As far as the problem domain is concerned, the financial portfolio optimization problem poses a challenge for several optimization techniques. The main goal is to efficiently construct portfolios of assets, as well as finding their optimal weights, which satisfy certain objective(s) and at the same time are subject to many constraints, in some cases.

In this study, two hybrid schemes are compared in terms of performance and convergence properties. Both techniques apply a genetic algorithm for asset selection. Regarding weight optimization the first algorithm uses a non-linear programming technique and the second one applies a continuous ant colony optimization (ACO) algorithm. The aim, in this study, is to provide some evidence concerning the

© Springer-Verlag Berlin Heidelberg 2011

performance of nature-inspired algorithms, i.e. ACO, in continuous optimization problems, i.e. weight optimization. Regarding the portfolio management problem, the objective is to maximize the Sortino ratio with a constraint on the tracking error volatility of the portfolio. This problem formulation incorporates a non-linear objective function, as well as a non-linear constraint. Finally, the main contribution of this study lies in providing proof, based on simulation results, regarding the convergence ability and performance of the aforementioned methodologies. What is more, this paper highlights, in a way, the usefulness of nature-inspired algorithms in dealing with hard optimization problems.

This paper is organized as follows. In section 1, some brief introductory comments are provided. In section 2, evidence from the literature, regarding selected studies which are related to the subject, are presented. In section 3, a brief analysis for the proposed algorithms is provided. In section 4, the mathematical formulation of the portfolio optimization problem is shown. In section 5, results from the computational study are provided and analyzed. Finally, in section 6, some useful concluding remarks and future research directions are shown.

2 Literature Review

In what follows, a brief analysis of related works is going to be presented. In any case, this analysis is not considered to be exhaustive. However, it could be representative of the specific field.

In [6], a hybrid scheme, combining simulated annealing with principles from evolutionary strategies, was applied to the classical Markowitz's portfolio optimization model. What is more, benchmark results from other two techniques, namely a simulated annealing algorithm and a special case of genetic simulated annealing method, were provided. Test data comprised of two sets, i.e. the German Index (DAX30) and the British Index (FTSE100). Results indicated that the incorporation of characteristics from evolutionary strategies may enhance the performance of local search algorithms. In [4], a genetic algorithm was combined with a quadratic programming approach, and this hybrid scheme was applied to a formulation of the passive portfolio management, where the objective was to minimize the tracking error volatility. What is more, authors provided benchmark results from a method, which applied random search for asset selection. Results indicated the superiority of the hybrid technique, and also that the optimal portfolios can efficiently track the benchmark index (Dutch AEX). In more related works, a hybrid scheme, combining an ACO algorithm for asset selection and the LMA algorithm for weight optimization, was applied to the active portfolio management problem under a downside risk objective [10]. The dataset comprised of stocks from the FTSE100 index. Also, two benchmark techniques were applied, namely an ACO algorithm with an encoding scheme and a random-selection algorithm for asset selection. Based on preliminary results, the performance of the hybrid scheme seems to be inferior compared to the algorithm, which combined an ACO with a weight encoding scheme. In [9], a particle swarm optimization algorithm was applied to three formulations of the active portfolio management problem, with a constraint to the tracking error volatility. The PSO algorithm aimed at selecting the assets, whereas a weight encoding heuristic was applied for weight optimization. Finally, in [2], an

ACO algorithm for asset selection was combined with a FA (firefly) algorithm for weight optimization. This hybrid scheme was applied to a portfolio optimization problem, where the objective was to maximize the Sortino ratio with a constrained on tracking error volatility. The dataset comprised of stocks from S&P's 500 index. Moreover, this scheme was compared with a Monte-Carlo algorithm and a financial heuristic. Results indicated the superiority of the proposed methodology.

All in all, the following remarks could be stated. Firstly, several techniques, which incorporate nature-inspired algorithms, have been applied to the portfolio optimization problem. Some examples are the application of particle swarm optimization and ant colony optimization algorithms in portfolio selection. Regarding the problem domain, several formulations of the portfolio optimization problem have been studied. Finally, as far as the main results from these studies are concerned, there is no clear conclusion of the superiority of hybrid nature-inspired methodologies. This could be indicative for more experimentation of these techniques.

3 Methodology

In this study, two hybrid intelligent methodologies are compared. The common component of these approaches is a real-valued genetic algorithm [3] which is applied in the discrete solution space in order to form high-quality portfolios of assets. In our approach, the genetic algorithm is comprised of three stages. At first the population (portfolios) is initialized in a random way. Afterwards, for a specified number of generations, *n-best* members of the population, based on their fitness value, are selected. As a next step, either crossover or mutation is applied to the selected members in order to form the descendants (new portfolios). Finally, the initial population is updated in a way that only high-quality portfolios, based on their fitness value, are included.

However, apart from constructing portfolios of assets, the capital invested in each of these assets should be determined. This is achieved by applying two different methodologies, specialized in continuous optimization. At the first hybrid, a non-linear programming methodology, based on mathematical principles[1], is applied so as to find a vector of weights which minimizes the given objective function under certain constraints [1]. At the second hybrid scheme, the ACO metaheuristic for continuous problems is used [8]. This type of continuous ACO works as follows. At first step, the population is randomly initialized. Then, for each generation, an artificial ant (vector of weight) is selected, based on its fitness value. In this step, a roulette wheel process is applied as well, in order to ensure that not always the best ant is selected (in this way, we avoid getting stuck in sub-optimal solutions). Then, for each of the *k-th* dimensions of the problem (where dimensionality corresponds to the cardinality of the problem), *pop* (where *pop* is the size of population) weights, which follow the normal distribution, are generated. As the mean of the distribution, the weight of the selected ant for this dimension is used. Also, as the standard deviation of the distribution, a metric concerning the deviation between the weights of other ants and the one of the selected ant, again for this dimension, is used. Finally, the pheromone update phase

[1] The local search algorithm that we use is based on the Levenberg – Marquardt method which combines the Gauss – Newton and the steepest descent method.

takes place, where high-quality solutions replace worst solutions in the population. In this formulation of the ACO algorithm, the pheromone matrix is replaced by an archive, where solutions are stored. The pheromone update process refers to the replacement of 'bad' solutions with 'good' ones, in this archive. So, in a sense, eventually all the population leads to good-quality solutions region.

In what follows, pseudocodes of the processes described above, are presented.

```
Function Genetic Algorithm
Parameter Initialization
Population Initialization
For i=1:generations
  Randomly choose genetic operator
  Apply genetic selection (choose n-best members of
  population)
  Apply Crossover or Mutation for producing new members
  Calculate weights/evaluate fitness value
  Adjust population in order to keep best members
End
```

Fig. 1. Genetic Algorithm

```
Function Continuous ACO
Parameter Initialization
Population Initialization
Define archive of solutions
For i=1:generations
  Apply Roulette Wheel for selection of an artificial ant,
  based on its rank in the solution archive (fitness value)
  For j=1:All_dimensions
    Define mean of normal distribution (value of artificial
  ant in j-th dimension)
    Define sigma of normal distribution (distance between
  each ant and the selected ant in j-th dimension)
    Sample pop-numbers randomly generated in normal
  distribution
  End
  Evaluate new solutions
  Update solution archive (keeping best solutions in it)
End
```

Fig. 2. Continuous ACO

4 Application Domain

As it was mentioned above, the portfolio optimization problem deals with finding a combination of assets, as well as the corresponding amount of capital invested in them, with the aim of optimizing a given objective function (investor's goal) under certain constraints. There are various formulations of the objective function, linear or not. Each one of them resides to a different type of problems.

Passive portfolio management is adopted by investors who believe that financial markets are efficient, i.e. it is impossible to consistently beat the market. So, the main objective is to achieve a similar level of returns and risk, as possible, of a certain benchmark. One passive portfolio management strategy is index tracking, i.e. construction of a portfolio, using assets from a universe of assets (like a stock index), with the attempt to reproduce the performance of the stock index itself [4]. In this study, a constraint on tracking error volatility, i.e. a measure of the deviation between the portfolio's and benchmark's return, is imposed.

The objective of the portfolio optimization problem is to maximize a financial ratio, namely the Sortino ratio [5]. The definition of the Sortino ratio is based on the preliminary work of Sharpe (Sharpe ratio) [7], who developed a reward-to-variability ratio. The main concept was to create a criterion that takes into consideration both assets' expected return and volatility (risk). However, in recent years, investors started to adopt the concept of "good volatility", which considers returns above a certain threshold, and "bad volatility", which considers returns below a certain threshold. A minimum dispersion of returns which fall below a certain threshold is desirable by investors. So, Sortino ratio considers only the volatility of returns, which fall below a defined threshold.

The mathematical formulation of the financial optimization problem is presented below:

$$Maximize\ Sortino\ Ratio = {E(r_p) - r_f}\big/{\theta_0(r_p)} \tag{1}$$

s.t.

$$\textstyle\sum_{i=1}^{N} w_i = 1 \tag{2}$$

$$-1 \leq w_i \leq 1 \tag{3}$$

$$\sqrt{Var(r_p - r_B)} \leq H \tag{4}$$

$$\mathrm{k} = \mathrm{N} \tag{5}$$

where,
$E(r_P)$, is the portfolio's expected return
r_f, is the risk-free return
$\theta_0(r_P)$, is the volatility of returns which fall below a certain threshold and equals

$$\theta_0(r_P) = \sqrt{\int_{-\infty}^{r_{th}} (r_{th} - r_P)^2 * f(r_P) dr_P} \tag{6}$$

w_i, is the percentage of capital invested in the ith asset
r_B, is the benchmark's daily return
r_{th}, is the threshold return
H, is the upper threshold for the tracking error volatility
$f(r_P)$, is the probability density function of the portfolio's returns.
k, is the number of assets of the portfolio (cardinality constraint)

It has to be mentioned that the constraint on the tracking error volatility was incorporated in the objective function using a penalty term.

5 Computational Study

In order to evaluate the performance of the hybrid schemes, a number of independent simulations were conducted. Due to the stochastic behavior of nature-inspired intelligent algorithms, statistical properties of the distribution of the independent runs can provide a better insight on the performance and abilities of these techniques. Data sample comprised of 93 daily returns for 49 companies listed in the FTSE/ATHEX index, as well as the index itself, for the time period 04/01/2010-29/05-2010. In this time period, the market was bearish.

In Table 2, both the settings for the parameters of the hybrid schemes and the optimization problem are shown. Selection of these configuration settings was based both on findings from previous simulation experiments and on the fact that they yielded satisfactory results.

Table 1. Parameter Settings

Parameters for Genetic Algorithm	
Population	200
Generations	30/50
Crossover Probability	0.90
Mutation Probability	0.35
N-best percentage	0.10
Parameters for Optimization Problem	
Cardinality[2]	10/20/30
Lower threshold for weights	-1
Upper threshold for weights	1
Iterations for non-linear programming	1
H	0.0080
Penalty term	0.8
Parameters for Continuous ACO	
Population	50
Generation	80
Evaporation rate	0.85
q[3]	0.1

As it can be observed from Table 2, the performance of the hybrid schemes was studied for various generations and cardinalities. In Table 3, some useful statistics are presented.

In order to have a good insight regarding the distribution of the fitness value in each case, percentiles of the distributions were provided for various confidence levels. The notion of this statistical measure can be described as follows. If percentile of X is a in 0.05 confidence level, then there is a probability of 5% that X will get values less than a. In a sense, it is required that the values of the percentile for a given

[2] Settings for the various cardinalities were common to both hybrid algorithms.

[3] Parameter which defines the influence of the solutions. Very small values means that only the best solutions are considered.

Table 2. Statistical results for hybrid schemes

k=10/gen=30	**Percentiles of distribution**				
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01[4]	1.9246	2.2555	2.4228	2.5859	2.8664
Hybrid Scheme 02[5]	1.8863	2.2209	2.3683	2.4939	2.7793
k=10/gen=50					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.0668	2.3724	2.5360	2.7354	3.1765
Hybrid Scheme 02	2.0995	2.3638	2.4849	2.5969	2.7861
k=10/gen=100					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.2492	2.5246	2.6533	2.8228	3.1622
Hybrid Scheme 02	2.1682	2.5186	2.5982	2.6872	2.9363
k=20/gen=30					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.3665	2.6317	2.8352	3.0143	3.4405
Hybrid Scheme 02	2.3355	2.7645	3.0017	3.2435	3.6756
k=20/gen=50					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.6089	2.8773	3.0781	3.3194	3.7955
Hybrid Scheme 02	2.6939	3.1301	3.3304	3.5233	3.9625
k=20/gen=100					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.8364	3.1448	3.4262	3.6725	3.9741
Hybrid Scheme 02	3.1430	3.4844	3.6689	3.8074	4.1844
k=30/gen=30					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.1771	2.3470	2.4681	2.6050	2.9030
Hybrid Scheme 02	2.4721	2.7015	2.8598	2.9621	3.3474
k=30/gen=50					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.3260	2.5367	2.6621	2.8104	3.0979
Hybrid Scheme 02	2.8266	3.2055	3.4233	3.6566	4.0059
k=30/gen=100					
	0.025	***0.25***	***0.50***	***0.75***	***0.975***
Hybrid Scheme 01	2.4642	2.6840	2.8641	3.0165	3.3468
Hybrid Scheme 02	3.3520	3.6608	3.8159	3.9968	4.3562

[4] Hybrid 01: Genetic Algorithm with LMA.

[5] Hybrid 02: Genetic Algorithm with Continuous ACO.

distribution should be quite large (large values mean that the distribution leans to the right) and also the difference between the $X_{0.05}$ and $X_{0.95}$ should be small at the same time. This ensures that our distribution leans to the right, and at the same time has no long left tail. For example, in the case of cardinality 10 and 30 generations, the percentiles of the distribution of the first hybrid scheme indicate that there is 2.5% probability that fitness value will take values less than 1.9246, whereas in the case of the second hybrid scheme there is 2.5% probability that the fitness value will take values below 1.8863. Practically, the left tail of the first hybrid's distribution is shorter than the second's, which is a desirable attribute.

Based on the above results, the following main points could be stated. In the case of the 10-asset cardinality problem, the first hybrid slightly outperformed the second one in all different generations. Regarding the 20-asset cardinality problem, the first hybrid outperformed the second one only in the case of 30 generations. In all other cases, as well in all cases of the 30-asset portfolio optimization problem, the second hybrid scheme yielded better results. This could be intuitive concerning the performance of the second hybrid scheme, which combines a genetic algorithm for asset selection and a continuous ACO algorithm for weight optimization. In essence, this metaheuristic performs better in high dimensions of the portfolio optimization problem, i.e. where the cardinality is large. In our situation, the market comprises of 49 assets. As a result, constructing efficient portfolios using almost half the number of assets in the market (for cardinality 30) is a remarkable result. In general, the outcomes of Table 3 indicate that a hybrid algorithm comprising of nature-inspired intelligent algorithm for continuous space optimization performs better than a numerical optimization algorithm in high dimensional instances of the portfolio optimization problem.

However, in order to obtain a better insight of the performance of the hybrid schemes, some heuristic rules are provided for benchmarking. More specifically, two financial rules are applied. Based on the first rule (Heuristic 01), equally-weighted portfolios of *k*-assets are formed. Assets are selected based on their fitness value (Sortino ratio). Based on the second rule (Heuristic 02), an equally-weighted portfolio is formed using all the assets in the market. Applying another simple rule (Heuristic 03), equally-weighted portfolios are constructed randomly, i.e. random selection of assets for these portfolios. Results are shown in Table 4.

Table 4. Results from various heuristics

	Fitness Value
k=10	
Heuristic 01	0.0270
Heuristic 03	-0.1823
k=20	
Heuristic 01	-0.0575
Heuristic 03	-0.2238
k=30	
Heuristic 01	-0.1291
Heuristic 03	-0.2056
Heuristic 02	-0.2109

Based on the results of Table 4, it is clear that simple heuristic rules come up with unsatisfactory results, in terms of fitness value.

As it can be seen from the results present above, hybrid intelligent schemes outperformed some simple financial heuristic rules. In a sense, these can considered as naïve benchmarks, which are not as sophisticated as the proposed hybrid metaheuristics. However, these rules may implement simple rules of thumb. In essence, hybrid schemes, which incorporate intelligent concepts and metaheuristic techniques, are able to yield better solution, due to the fact that their searching strategy can handle complex solution spaces.

6 Conclusions

In this study, two hybrid metaheuristics were briefly analyzed and compared on a NP-hard optimization problem, i.e. the financial portfolio management. The aim of this work was to provide some evidence regarding the capabilities of a hybrid scheme, which combines a genetic algorithm and a nature-inspired metaheuristics (ACO). Moreover, the main motivation of employing this kind of hybrid scheme lies on the great potential of nature-inspired algorithms in searching complex continuous space efficiently.

Experimental results highlight the good performance of the proposed hybrid scheme in high dimensionality problems, as compared to the benchmark hybrid scheme which combines a genetic algorithm with a LMA technique. Moreover, the proposed hybrid algorithm yielded better results compared to simple rules of thumb. As far as the financial implications of this study are concerned, it seems that, in a market (universe of stocks) with 49 stocks, constructing portfolios with 30 assets (almost half the number of the total stocks in the market) yields good results. What is more, it can be observed that as the cardinality of the portfolio rises, the fitness value (Sortino ratio-the investment goal) increases.

Regarding the performance of the hybrid algorithm which combines the genetic algorithm with the ACO metaheuristic, it could be stated that the searching ability of the continuous technique has great potential, compared to the LMA algorithm.

As far as the future research is concerned, the issue of investigating other nature-inspired hybrid algorithms is quite promising. Another important aspect is the application of more intelligent (meta)heuristic rules for benchmarking, i.e. rules from financial experts. Last but not least, it could be interesting to apply the aforementioned hybrid techniques in larger stock markets in order to analyze the performance of the algorithm in higher cardinalities.

References

1. Byrd, R.H., Gilbert, J.C., Nocedal, J.: A trust region method based on interior point techniques for nonlinear programming. Mathematical Programming 89(1), 149–185 (2000)
2. Giannakouris, G., Vassiliadis, V., Dounias, G.: Experimental Study on a Hybrid Nature-Inspired Algorithm for Financial Portfolio Optimization. In: Konstantopoulos, S., Perantonis, S., Karkaletsis, V., Spyropoulos, C.D., Vouros, G. (eds.) SETN 2010. LNCS (LNAI), vol. 6040, pp. 101–111. Springer, Heidelberg (2010)

3. Holland, J.H.: Genetic Algorithms. Scientific American, 66–72 (1992)
4. Jeurissen, R., Berg, J.: Optimized index tracking using a hybrid genetic algorithm. In: IEEE Congress on Evolutionary Computation, pp. 2327–2334 (2008)
5. Kuhn, J.: Optimal risk-return tradeoffs of commercial banks and the suitability of profitability measures for loan portfolios. Springer, Berlin (2006)
6. Maringer, D., Kelleler, H.: Optimization of Cardinality Constrained Portfolios with a Hybrid Local Search Algorithm. OR Spectrum 25, 481–495 (2003)
7. Sharpe, W.F.: The Sharpe ratio. Journal of Portfolio Management, 49–58 (1994)
8. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research 185, 1155–1173 (2008)
9. Thomaidis, N.S., Angelidis, T., Vassiliadis, V., Dounias, G.: Active Portfolio Management with Cardinality Constraints: An Application of Particle Swarm Optimization. New Mathematics and Natural Computation, Working Paper (2008)
10. Vassiliadis, V., Thomaidis, N., Dounias, G.: Active Portfolio Management under a Downside Risk Framework: Comparison of a Hybrid Nature – Inspired Scheme. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 702–712. Springer, Heidelberg (2009)

Genetic Defect Based March Test Generation for SRAM

Stefano Di Carlo, Gianfranco Politano, Paolo Prinetto, Alessandro Savino, and Alberto Scionti

Politecnico di Torino, Control and Computer Engineering Department, Torino I-10129, Italy
{stefano.dicarlo,gianfranco.politano,paolo.prinetto, alessandro.savino,alberto.scionti}@polito.it
http://www.testgroup.polito.it

Abstract. The continuos shrinking of semiconductor's nodes makes semiconductor memories increasingly prone to electrical defects tightly related to the internal structure of the memory. Exploring the effect of fabrication defects in future technologies, and identifying new classes of functional fault models with their corresponding test sequences, is a time consuming task up to now mainly performed by hand. This paper proposes a new approach to automate this procedure exploiting a dedicated genetic algorithm.

Keywords: Memory testing, march test generation, defect based testing.

1 Introduction

Semiconductor memories have been used for long time to push the state-of-the-art in the semiconductor industry. The Semiconductor Industry Association (SIA)[1] forecasts that in the next 15 years up to 95% of the entire chip area will be dedicated to memory blocks. Precise fault modeling and efficient test design are therefore pivotal to keep test cost and time within economically acceptable limits.

Functional Fault Models (FFMs) coupled with efficient test algorithms such as march tests (the reader may refer to [1] to understand the concept of march test) have been so far enough to deal with emerging classes of memory defects [1]. FFMs do not depend on the specific memory technology and allow automation of test sequences generation [3]. Exploring the effect of fabrication defects in future technologies, and identifying new classes of FFMs with their relevant test sequences, is a time consuming task up to now mainly performed by hand. However, the continuos shrinking of semiconductor's nodes makes semiconductor memories increasingly prone to electrical defects tightly related to the internal structure of the memory [8,7]. Automating the analysis of these defects is mandatory to guarantee the quality of next generation memory devices.

[1] http://www.itrs.net/

© Springer-Verlag Berlin Heidelberg 2011

Automatic defect level memory test generation is an area of test generation and diagnosis that still needs to be fully explored. Cheng et al. [4] presented FAME, a fault-pattern based memory failure analysis framework that applies diagnosis-based fault analysis to narrow down potential causes of failure for a specific memory architecture. Results of the analysis are then used to optimize a given test sequence (march test) considering only the observed faults. Similarly, Al-Ars at al.[2] proposed a framework for fault analysis and test generation in DRAMs that uses Spice to model the memory under test and the target defects. Spice simulations are used to perform fault analysis starting from well known test algorithms available in literature. The main drawback of these methods is that they allow the optimization of existing test sequences rather than the generation of new and optimized set of stimuli.

This paper tries to overcome these problems proposing a software framework for defect level march test generation . The generation process exploits a genetic algorithm able to highlight faulty behaviors in a defective memory and to generate the related test sequences by means of electrical simulations. The use of a genetic algorithm allows an efficient exploration of a huge space of march test alternatives guaranteeing high defect coverage.

2 Generation Framework Architecture

Figure 1 overviews the architecture of the proposed framework. The core block of the system is the Genetic March Test Generator (GMTG), a genetic algorithm used to drive the march test generation process.

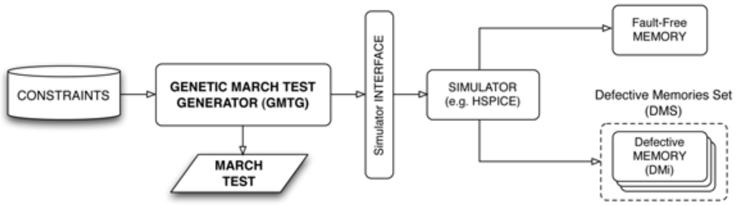


Fig. 1. March test generation framework

We use electrical Spice models to precisely model the memory behavior and the characteristics of the fabrication process (fault-free memory of Figure 1). In a similar way, memory defects are modeled as electrical components (e.g., resistors, parameter changes, etc.) on the fault-free memory obtaining a collection of defective memories (DM_i) named Defective Memory Set (DMS). Each defective memory is characterized by a single defect.

The GMTG operates trying to highlight erroneous behaviors caused by the inserted defects, and providing a march test for their detection. The comparison between the fault-free and the defective memory models is performed by analyzing their electrical simulations (simulator block of Figure 1) when the target test sequence is applied. To allow adaptation to different types of memories, description levels, description languages, and simulators, an interface layer is placed between the GMTG and the simulator to virtualize the specific commands and results format.

3 Genetic March Test Generator

The following pseudocode describes the way the GMTG algorithm works while generating a march test for a specific set of defects.

```
GMTG (): begin
 1:     solution = "";
 2:     foreach (DM_i in DMS) {
 3:       generation=0; generate initial population based on current solution;
 4:       simulate (DM_i , population);
 5:       while (generation < MAX_GEN) {
 6:         coverage = check_coverage(solution,DM_i);
 7:         if (coverage) break;
 8:         evolve (population, offspringsize)
 9:         validate (population);
10:         simulate (DM_i , population);
11:         evaluate_fitness(population);
12:         solution = update (population,solution);
13:         generation ++;}
14:       if (generation==MAX_GEN) exit_without_solution();}
15:     show_solution (solution);
end
```

The algorithm starts with an empty solution (row 1). Each defect DM_i is examined separately with an iterative process (row 2). This slightly modifies the algorithm structure w.r.t. traditional genetic algorithms. For each defect a random population of individuals is generated. Individuals of the population (chromosomes) represents candidate march tests, and the individual with higher fitness represents the candidate solution. When generating the population for a new defect, all new individuals will contain the genes of the current solution plus new additional genes to guarantee the coverage of the already analyzed defects. Rows 5 to 13 are the actual genetic process with each iteration representing the evolution of the population from a generation to the next one. First the coverage of the current solution w.r.t. the target defect is evaluated (row 6). If the current solution already detects a faulty behavior in the defective model the generation stops and moves to the next model (row 7). If not, the population is evolved applying different genetic operators explained later in this section (row 8). `offspring_size` represents the number of individuals to substitute in the current population. The new population is validated to guarantee that each chromosome correctly represents a march test (row 9). Chromosomes that do not pass the validation can be either discarded or genetically modified to fit the constraints. The population is then simulated w.r.t. the target defect (row 10), the fitness of each individual is computed (row 11), the new candidate solution is selected (row 12) and the process continues. If the evolution reaches a maximum number of iterations without identifying a suitable solution the generation fails and the algorithm ends (row 14).

3.1 Chromosome Encoding

The proposed genetic algorithm works with chromosomes representing candidate march tests. According to [10] there are six Degrees Of Freedom (DOF) that can be exploited to increase the detection capabilities of march tests. These DOFs are considered in our chromosome encoding to enhance the efficiency of the GMTG. Each chromosome is composed of a sequence of genes representing basic memory operations used to build a march test. Each gene is encoded using a binary string including the following basic fields:

- *start marker* (1 bit): when asserted, it denotes the beginning of a new march element within the current gene;
- *addressing order* (1 bit): defined in correspondence of the beginning of a march element to identify its addressing order (1: direct addressing order $\Uparrow$, 0: inverse addressing order $\Downarrow$);
- *stop marker* (1 bit): when asserted denotes that the current gene concludes a march element;
- *operation*: a sequence of bits encoding the memory operation to apply. The list of available memory operations depends on the target memory (basic operations we considered are write, read, and idle). During the generation process, the simulator interface (see Figure 1) translates each operation into the correct sequence of signals for the memory;
- *data*: defined in case of write operations, it represents the value to write into the memory (0 or 1 in case of single bit memories);
- *addressing sequence*: is the sequence of addresses associated with the direct and reverse addressing order. This field exploits the first two degrees of freedom proposed in [10]:(i) the addressing sequence can be freely chosen as long as all addresses occur exactly once and the sequence is reversible (DOF1); (ii) the addressing sequence for initialization can be freely chosen as long as all addresses occur at least once (DOF2). In this work we consider two possible sequences: (i)
  - *column mode* $\langle$c$\rangle$:each memory cell in a given row of the memory cell array is scanned before moving to the next row;
  - *row mode* $\langle$r$\rangle$: each cell in a given column of the memory cell array is scanned before moving to the next column.

  Additional and more complex addressing sequences can be defined and added to the encoding schema to increase the space of possible solutions and to enhance the detection capabilities of the generated algorithms;
- *data pattern*: this field allows to exploits another degree of freedom defined in [10]: the data within a read/write operation does not need to be the same for

all memory addresses as long as the detection probabilities for basic faults are not affected (DOF4). Basically this DOF allows to change the data written into the memory while moving to the different cells of the array. We consider four possible data patterns (solid ⟨sol⟩: all cells are written with the same value, checkboard ⟨ckb⟩: cells are written with alternate values, alternate row ⟨alr⟩: rows of the memory are filled with alternate values, and alternate column ⟨alc⟩: columns of the memory are filled with alternate values).

Each chromosome encodes at least a write operation needed to initialize the memory array with a well known value and possibly sensitize a faulty behavior and a read operation to observe the faulty behavior. The number of sensitizing operations can be then incremented to deal with more complex defects or combination of defects.

3.2 Population Validation

During the evolution from a generation to the next one, the application of the genetic transformations may lead to chromosomes with undesired properties, i.e., chromosomes that do not represent valid march tests. To avoid this situation, when new individuals are generated their structure must be validated. Incorrect individuals are not killed but whenever possible their structure is healed applying a set of transformations. These transformations work on the start and stop markers of genes based on the following rules:

- if a gene has the stop marker asserted, the start marker of the next gene of the chromosome must be asserted;
- if a gene has a start marker asserted, the stop marker of the previous gene of the chromosome must be asserted;
- the start marker of the first gene and the stop marker of the last gene of a chromosome must always be asserted.

We decided to introduce these transformations instead of simply discarding individuals with erroneous genetic content to avoid discarding genetic material that may contain interesting characteristics for the final solution.

3.3 Fitness Function

The fitness function is the key element used to drive the evolution process and in particular to select those chromosomes that most likely lead to valid solutions of the problem. The idea is to identify a function that privileges the ability of an individual of sensitizing faulty behaviors, i.e., the ability of producing different electrical signals at the nodes of the fault-free and defective memories. This imposes to define a fitness function able to evaluate analog differences among signals.

The computation of the fitness is based on the concept of *probe nodes*, i.e., I/O nodes of a memory cell (i.e., bit lines) or output nodes of a sense amplifier. The electrical signals (i.e., voltage) produced at each probe node of the target memory during the electrical simulation of the test sequence associated to a chromosome are traced. These values are then analyzed and combined into a value of fitness according to Eq. 1:

$$f(x) = \sum_{t=0}^{T_{max}} \sum_{i=0}^{N_{probe}-1} D_{i,t} \tag{1}$$

where x is the target chromosome, t the simulation time and N_{probe} the number of analyzed probe nodes. $D_{i,t}$ represents the absolute value of the difference between the voltage at probe node i in the fault-free memory and the voltage at the probe node i in the defective memory, at simulation time t. These differential values are combined together considering all probe nodes and simulation times by the two sums of Eq. 1. The proposed function has two main drawbacks:

– it can easily lead to populations with very small differences between the individuals (premature convergence). This actually turns the evolution process into a random selection among chromosomes reducing the efficiency of the genetic approach;
– it can produce among a high number of similar individuals, a single chromosome (super chromosome) with fitness much higher than all the remaining ones. This is again negative since the evolution will be completely polarized by this chromosome and the space of solutions will not be correctly explored.

To leverage these problems, we introduced a linear normalization able to correctly distribute the fitness values. The population is sorted by decreasing fitness values. Chromosomes in the sorted list receive a new scaled fitness $f_s(x)$ according to Eq. 2.

$$f_s(x) = C - n \cdot L \tag{2}$$

where C is a constant value, L represents the linear normalization step (a parameter of the method), and n is the position of the chromosome in the sorted list.

3.4 Evolution

During the generation process, when the current solution does not provide the required defect coverage the current population is evolved substituting a set of individuals with new ones with different characteristics. In our framework, in addition to traditional genetic operators (e.g., crossover [5]) we apply additional rules during the evolution. First of all, if a certain sequence of genes in the chromosome has been used to detect a certain defect, it cannot be modified while analyzing a new defect. This in turns requires to add new genes to the sequence in order to have a certain degree of freedom in the modification of the individuals. This is performed introducing a new genetic operator named *increase_chromosome_length* able to increase by one the number of genes composing the chromosomes of the population. The problem in this case is the selection of the type of gene to insert and, in particular, the type of memory operation the new gene has to encode. Based on the fact that some fault models (i.e., dynamic faults) are sensitized by long sequences of identical operations, the approach we adopted inserts new genes repeating the last operation in the sensitizing sequence of each chromosome. Moreover, the new gene is always inserted as part of the last march element.

3.5 Coverage Conditions

For each defective memory model DM_i the GMTG ends the generation process when either a chromosome able to sensitize and detect a faulty behavior appears in the population, or a maximum computation effort is reached. Every time a new solution is identified the electrical simulations of the fault-free and target defective memories are compared to identify if the given test sequence is able to detect a new erroneous behavior. In particular, the analysis focuses on the portions of the simulation (samples) corresponding to the genes encoding read operations. The logic value returned by the observations is calculated (taking into account the electrical parameters of the target memory) for the fault-free and the defective memory. When the two values differ, a faulty behavior is detected and the generation ends.

4 Experimental Results

The capability of the proposed framework has been validated on a 3×3 SRAM consisting of an array of 9 classical 6 transistors cells implemented using the 130nm Predictive Technology Model[2]. The use of a reduced 3×3 matrix allows to maintain the simulation time under control. Nevertheless, this simplification still allows to obtain realistic results since it has been demonstrated in [1] that defects are usually localized in a range of a few cells. Electrical simulations have been performed using the commercial HSPICE$^{\mathrm{TM}}$ simulator while the GMTG has been implemented in about 3,500 lines of C code executed on a 1.8GHz AMD TURION$^{\mathrm{TM}}$ laptop equipped with 1GB of RAM and running the Linux operating system. A preliminary set of experiments allowed us to derive the set of tuning parameters for the GMTG[3].

Figure 2 (a) and Figure 2 (b) propose the architecture of a single memory cell including our target collection of defects. Figure 2 (a) proposes seven typical resistive defect locations deeply analyzed in literature [6,7], whereas Figure 2 (b) proposes the set of short defects analyzed in [9]. During our experiments these defects have been injected in the first cell of the memory matrix.

Table 1 shows the results obtained by considering both resistive defects (DFT) and shorts (DFS) in isolation. For resistive defects, the minimum value of resistance able to produce a faulty behavior (Rmin) is reported while for shorts, we considered a resistive value of 1.0Ω. All simulations have been performed at a temperature of 27°C. For each defect the generated test sequence and the generation time are also provided. The obtained results show that we have been

[2] `http://www.eas.asu.edu/~ptm`

[3] MAX_GEN=200, population=10, offspring_size=5, C=250, L=25.

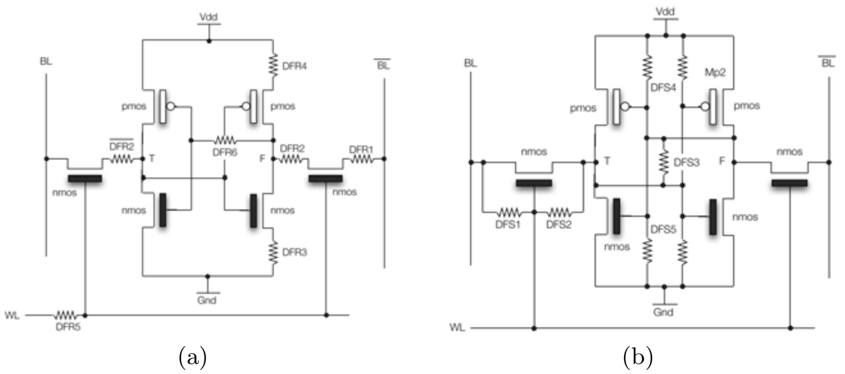


Fig. 2. Memory cell with target resistive defects (a) and short defects (b)

able to automatically generate dedicated march tests for resistive defects DFR1, DFR2, DFR3, DFR5, DFR6, $\overline{DFR2}$ and for all short defects with a very low effort in terms of execution time. The generated test sequences are consistent with the studies performed in [6] for resistive defects and in [9] for shorts, thus confirming the effectiveness of the proposed approach.

Table 1. March tests for single defects. Simulations have been performed using $T = 27°$C. Defects are measured in MΩ, while the execution time is expressed in s.

DFR	**Rmin**	**Time**	**Test Sequence**	**DFS**	**Time**	**Test Sequence**
DFR1	0.025	178	$\langle c\rangle\langle ckb\rangle$ $\{\Uparrow (W0, W1, R1);\}$	DFS1	180	$\langle c\rangle\langle ckb\rangle$ $\{\Uparrow (W0, W1, R1);\}$
DFR2	0.020	175	$\langle r\rangle\langle sol\rangle$ $\{\Uparrow (W0, W1, R1);\}$	DFS2	180	$\langle r\rangle\langle sol\rangle$ $\{\Uparrow (W0, W1, R1);\}$
DFR3	0.007	171	$\langle c\rangle\langle alc\rangle$ $\{\Uparrow (W1, R1, R1);\}$	DFS3	180	$\langle c\rangle\langle sol\rangle$ $\{\Uparrow (W0, W1, R1);\}$
DFR4[4]	64.0	416	$\langle c\rangle\langle ckb\rangle$ $\{\Uparrow (W0, R0); \Uparrow (R0);\}$	DFS4	180	$\langle r\rangle\langle sol\rangle$ $\{\Uparrow (W0, W1, R1);\}$
DFR5	2.0	177	$\langle r\rangle\langle alr\rangle$ $\{\Uparrow (W1, W0, R0);\}$	DFS5	180	$\langle r\rangle\langle alr\rangle$ $\{\Uparrow (W0, W1, R1);\}$
DFR6	2.0	168	$\langle c\rangle\langle alc\rangle$ $\{\Uparrow (W1, W0, R0);\}$	-	-	
$\overline{DFR2}$	2.0	150	$\langle c\rangle\langle alc\rangle$ $\{\Uparrow (W1, W0, R0);\}$	-	-	

Table 2. March tests for multiple defects. Execution time is expressed in s.

#Exp	**Time**	**Test sequence**
EXP1	510	$\langle r\rangle\langle sol\rangle$ $\{\Uparrow (W0, W1, R1); \Downarrow (W0, R0);\}$
EXP2	2050	$\langle c\rangle\langle sol\rangle$ $\{\Uparrow (W0, W1, R1, W0, R0);\}$
EXP3	2062	$\langle c\rangle\langle ckb\rangle$ $\{\Uparrow (W0, W1, R1); \Downarrow (W0, W0); \Uparrow (R0); \Uparrow (W1); \Uparrow (R1); \Downarrow (R1);\}$

Slightly more complex is the situation for DFR4. Experiments performed at simulation temperature of 27°C were not able to identify any faulty behavior. This is again coherent with the results of [6]. We thus performed different experiments changing the operational temperature. By setting the temperature to 125°C and the defect size to 64.0MΩ, we have been able to produce a defective behavior and to obtain a corresponding test sequence as shown in Figure 3.

[4] Defect DFR4 has been simulated using $T = 125°$C.

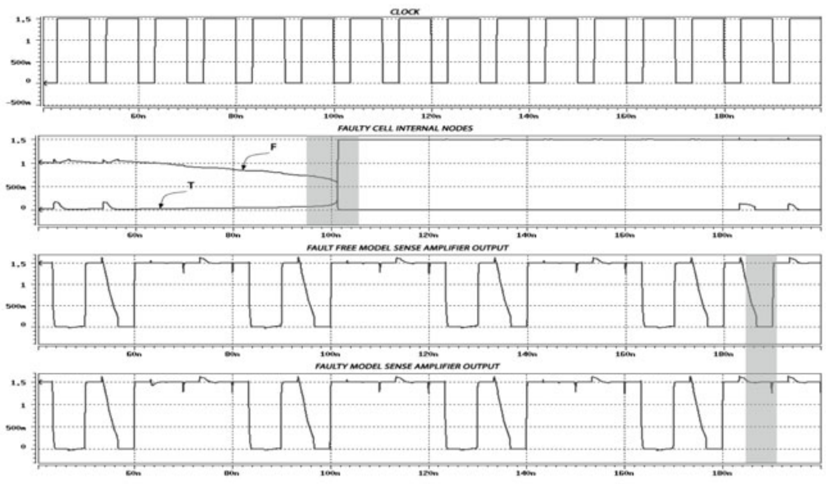


Fig. 3. Electrical simulation for DFR4 with $T = 125°C$ and $R = 64.0M\Omega$

The result obtained with DFR4 is particularly interesting. Looking at the results proposed in [6], the authors identified for the same type of defect, injected in their target memory, a dynamic faulty behavior instead of a data retention fault. Once again, this result stresses the importance of resorting to an automatic tool able to automatically generate test sequences customized on the target memory.

In addition to the previous experiments we performed a set of three experiments with groups of defects summarized by the results of Table 2. EXP1 considers a target defect list composed of DFR2 and $\overline{DFR2}$. According to the results of Table 1 these two defects introduce a Transition Fault into the memory. Looking at the generated test sequence in Table 2 we exactly obtained a march test able to detect the Transition Fault. The second experiment (EXP2) considers the same collection of defects of EXP1 plus DFS3. Again the result of the three defects can be modeled as a Transition Fault and the generated march test is able to detect this type of fault. Finally, EXP3 adds DFR3 to the defects considered in EXP2. The generated sequence is able to detect the transition fault introduced by the first three defects and also the read fault introduced by DFR3. However, in this case, it is clear that the generated sequence contains redundant operations. This situation is a direct consequence of the use of a genetic approach to generate the test. Nevertheless, a post elaboration can be applied in order to optimize the generated sequences.

5 Conclusion

This paper proposed a set of preliminary results toward the solution of the problem of defect based automatic march test generation. The proposed approach is based on a genetic algorithm able to identify faulty behaviors in a defective memory and to generate the corresponding test sequences. The use of a genetic approach allows an efficient exploration of a huge space of march test alternatives, guaranteeing high defect coverage and thereby reducing the time test engineers need to explore test alternatives. Experimental results show the effectiveness of the approach that proved to be able to reproduce results of previous studies with an acceptable execution time and without human intervention.

References

1. Al-Ars, Z., van de Goor, A.: Static and dynamic behavior of memory cell array spot defects in embedded drams. IEEE Trans. on Comp. 52(3), 293–309 (2003)
2. Al-Ars, Z., Hamdioui, S., Mueller, G., van de Goor, A.: Framework for fault analysis and test generation in drams. In: Proceedings Design, Automation and Test in Europe, pp. 1020–1021 (2005)
3. Benso, A., Bosio, A., Di Carlo, S., Di Natale, G., Prinetto, P.: March test generation revealed. IEEE Trans. Comput. 57(12), 1704–1713 (2008)
4. Cheng, K.L., Chih-Wea, W., Jih-Nung, L., Yung-Fa, C., Chih-Tsun, H., Cheng-Wen, W.: Fame: a fault-pattern based memory failure analysis framework. In: International Conference on Computer Aided Design, ICCAD 2003, November 9-13, pp. 595–598 (2003)
5. Davis, L.: Handbook of genetic algorithms. Van Nostrand Reinhold, New York (1991)
6. Dilillo, L., Girard, P., Pravossoudovitch, S., Virazel, A., Borri, S., Hage-Hassan, M.: Resistive-open defects in embedded-sram core cells: analysis and march test solution. In: IEEE 13th Asian Test Symposium, ATS 2004, November 15-17, pp. 266–271 (2004)
7. Dilillo, L., Girard, P., Pravossoudovitch, S., Virazel, A., Hage-Hassan, M.: Data retention fault in sram memories: analysis and detection procedures. In: 23rd IEEE VLSI Test Symposium (VTS 2005), May 1-5, pp. 183–188 (2005)
8. Hamdioui, S., Al-Ars, Z., van de Goor, A.: Opens and delay faults in cmos ram address decoders. IEEE Trans. Comput. 55(12), 1630–1639 (2006)
9. Huang, R.F., Chou, Y.F., We, C.W.: Defect oriented fault analysis for sram. In: Proceedings of the 12th Asian Test Symposium, pp. 1–6 (2003)
10. Niggemeyer, N., Redeker, M., Ottersted, J.: Integration of non-classical faults in standard march tests. In: IEEE International Workshop on Memory Technology, Design and Testing, MTDT 1998, August 24-26, pp. 91–96 (1998)

Improving ESOP-Based Synthesis of Reversible Logic Using Evolutionary Algorithms

Rolf Drechsler, Alexander Finder, and Robert Wille

Institute of Computer Science, University of Bremen, Bremen, Germany
{drechsle,final,rwille}@informatik.uni-bremen.de,
http://www.informatik.uni-bremen.de/agra/eng

Abstract. Reversible circuits, i.e. circuits which map each possible input vector to a unique output vector, build the basis for emerging applications e.g. in the domain of low-power design or quantum computation. As a result, researchers developed various approaches for synthesis of this kind of logic. In this paper, we consider the ESOP-based synthesis method. Here, functions given as *Exclusive Sum of Products* (ESOPs) are realized. In contrast to conventional circuit optimization, the quality of the resulting circuits depends thereby not only on the number of product terms, but on further criteria as well. In this paper, we present an approach based on an evolutionary algorithm which optimizes the function description with respect to these criteria. Instead of ESOPs, *Pseudo Kronecker Expression* (PSDKRO) are thereby utilized enabling minimization within reasonable time bounds. Experimental results confirm that the proposed approach enables the realization of circuits with significantly less cost.

Keywords: Evolutionary Algorithms, Reversible Logic, Synthesis, Exclusive Sum of Products, Pseudo Kronecker Expressions, Optimization.

1 Introduction

Reversible logic [11,1,21] realizes n-input n-output functions that map each possible input vector to a unique output vector (i.e. bijections). Although reversible logic significantly differs from traditional (irreversible) logic (e.g. fan-out and feedback are not allowed), it has become an intensely studied research area in recent years. In particular, this is caused by the fact that reversible logic is the basis for several emerging technologies, while traditional methods suffer from the increasing miniaturization and the exponential growth of the number of transistors in integrated circuits. Researchers expect that in 10-20 years duplication of transistor density every 18 months (according to *Moore's Law*) will come to a halt (see e.g. [24]). Then, alternatives are needed. Reversible logic offers such an alternative as the following applications show:

– *Reversible Logic for Low-Power Design*
  Power dissipation and therewith heat generation is a serious problem for today's computer chips. Landauer and Bennett showed in [11,1] that (1) using

© Springer-Verlag Berlin Heidelberg 2011

traditional (irreversible) logic gates always leads to energy dissipation regardless of the underlying technology and (2) that circuits with zero power dissipation must be information-lossless. This holds for reversible logic, since data is bijectively transformed without losing any of the original information. Even if today energy dissipation is mainly caused by non-ideal behaviors of transistors and materials, the theoretically possible zero power dissipation makes reversible logic quite interesting for the future. Moreover, in 2002 first reversible circuits have been physically implemented [5] that exploit these observations in the sense that they are powered by their input signals only and did not need additional power supplies.

– *Reversible Logic as Basis for Quantum Computation*
  Quantum circuits [14] offer a new kind of computation. Here, qubits instead of traditional bits are used that allow to represent not only 0 and 1 but also a superposition of both. As a result, qubits can represent multiple states at the same time enabling enormous speed-ups in computations. Even if research in the domain of quantum circuits is still at the beginning, first quantum circuits have already been built. Reversible logic is important in this area, because every quantum operation is inherently reversible. Thus, progress in the domain of reversible logic can directly be applied to quantum logic.

Further applications of reversible logic can be found in the domain of optical computing [3], DNA computing [1], and nanotechnologies [12].

Motivated by these promising applications, various synthesis approaches for reversible logic have been introduced in the past. They rely on different function representations like truth-tables [13], permutations [16], BDDs [23], or positive-polarity Reed-Muller expansion [9].

In the following, we focus on a method based on *Exclusive Sum of Products* (ESOPs) representations [6]. Here, the fact is exploited that a single product of an ESOP description directly corresponds to an appropriate reversible gate. The cost of the respective gates strongly depends thereby on the properties of the products. Accordingly, the quality of the resulting circuits relies not only on the number of product terms of the ESOP, but on further criteria as well. This is different to conventional logic optimization and, thus, requires an appropriate treatment.

In this paper, an approach is introduced which optimizes a given *Pseudo Kronecker Expression* (PSDKRO) with respect to these criteria. PSDKROs represent a subclass of ESOPs enabling minimization within reasonable time bounds. In order to optimize the PSDKROs, the evolutionary algorithm introduced in [7] is utilized. We describe how this algorithm can be extended to address the new cost models. Experimental results show that this leads to significant improvements in the costs of the resulting circuits. In fact, in most of the cases, the respective costs can be decreased by double-digit percentage points.

The remainder of this paper is structured as follows. The next section introduces the necessary background on reversible circuits as well as on ESOPs and PSDKROs. Afterwards, the ESOP-based synthesis method is briefly reviewed in Section 3. Section 4 describes the proposed optimization approach. Finally, experimental results are presented in Section 5 and conclusions are drawn in Section 6, respectively.

2 Background

To keep the paper self-contained, this section briefly reviews the basic concepts of reversible logic. Afterwards, ESOPs and PSDKROs are introduced.

2.1 Reversible Circuits

Reversible circuits are digital circuits with the same number of input signals and output signals. Furthermore, reversible circuits realize bijections only, i.e. each input assignment maps to a unique output assignment. Accordingly, computations can be performed in both directions (from the inputs to the outputs and vice versa).

Reversible circuits are composed as cascades of reversible gates. The *Toffoli gate* [21] is widely used in the literature and also considered in this paper. A Toffoli gate over the inputs $X = \{x_1, \ldots, x_n\}$ consists of a (possibly empty) set of *control lines* $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ and a single *target line* $x_j \in X \setminus C$. The Toffoli gate inverts the value on the target line if all values on the control lines are assigned to 1 or if $C = \emptyset$, respectively. All remaining values are passed through unaltered.

Example 1. Fig. 1(a) shows a Toffoli gate drawn in standard notation, i.e. control lines are denoted by ●, while the target line is denoted by $\oplus$. A circuit composed of several Toffoli gates is depicted in Fig. 1(b). This circuit maps e.g. the input 101 to the output 010 and vice versa.

Since the number of gates in a cascade is a very poor measure of the cost of a reversible circuit, different metrics are applied (sometimes depending on the addressed technology). In this work, we consider quantum cost and transistor cost. While the quantum cost model estimates the cost of the circuit in terms of the number of elementary quantum gates [14], the transistor cost model estimates the cost of the circuit in terms of the number of CMOS transistors [20]. Both metrics define thereby the cost of a single Toffoli gate depending on the number of control lines. More precisely:

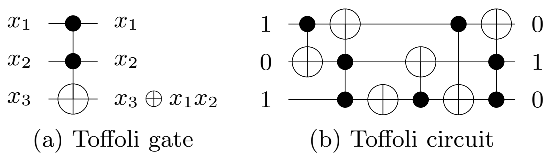


Fig. 1. Toffoli gate and Toffoli circuit

Table 1. Cost of reversible circuits

(a) Quantum cost

c	$(n-c+1) \geq$	cost
0		1
1		1
2		5
3		13
4	2	26
4	0	29
5	3	38
5	1	52
5	0	61
6	4	50
6	1	80
6	0	125
7	5	62
7	1	100
7	0	253
8	6	74
8	1	128
8	0	509
9	7	86
9	1	152
9	0	1021
> 9	$c-2$	$12(c+1)-34$
> 9	1	$24(c+1)-88$
> 9	0	$2^{c+1}-3$

(b) Transistor cost

s	cost
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56
8	64
9	72
10	80
> 10	$8 \cdot s$

– *Quantum cost model:* The quantum cost of a Toffoli gate is given in Table 1(a) (using the calculations according to [17]), where c denotes the number of control lines for the gate and n denotes the number of circuit lines. Note that the quantum cost depend not only on the number c of control lines, but also on the number $(n-c+1)$ of lines neither used as control line or target lines. The more lines are not in the set of control lines, the cheaper the respective gate can be realized.
– *Transistor cost model:* The transistor cost of Toffoli gate increases linearly with $8 \cdot s$ where s is the number of control lines in the gate (see Table 1(b)).

The cost of a circuit is the sum of the costs of the individual gates. For example, the gate shown in Fig. 1(b) has quantum cost of 14 and transistor cost of 56.

2.2 Exclusive Sum of Products and Pseudo Kronecker Expressions

Exclusive Sum of Products (ESOPs) are two-level descriptions of Boolean functions. Each ESOP is composed of various conjunctions of literals (called *products*). A *literal* either is a propositional variable or its negation. To form the ESOP, all products are combined by Exclusive ORs. That is, an ESOP is the most general form of two-level AND-EXOR expressions.

Since the minimization of general ESOPs is computationally expensive, several restricted subclasses have been considered in the past, e.g. *Fixed Polarity Reed-Muller Expressions* (FPRMs) [15] and *Kronecker Expressions* (KROs) [4]. As an interesting alternative, *Pseudo Kronecker Expressions* (PSDKROs) have been proposed, since the resulting forms are of moderate size, i.e. close to ESOPs, and the minimization process can be handled within reasonable time bounds. The following inclusion relationship can be stated for ESOPs and PSDKROs: $FPRM \subseteq KRO \subseteq PSDKRO \subseteq ESOP$.

Let f_i^0 (f_i^1) denote the *cofactor* of the Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ with $x_i = 0$ ($x_i = 1$) and $f_i^2 := f_i^0 \oplus f_i^1$, where $\oplus$ is the Exclusive OR operation. Then, f then can be represented by:

$$f = \overline{x}_i f_i^0 \oplus x_i f_i^1 \qquad \textit{(Shannon; abbr. S)} \qquad (1)$$
$$f = f_i^0 \oplus x_i f_i^2 \qquad \textit{(positive Davio; abbr. pD)} \qquad (2)$$
$$f = f_i^1 \oplus \overline{x}_i f_i^2 \qquad \textit{(negative Davio; abbr. nD)} \qquad (3)$$

A PSDKRO is obtained by applying either S, pD, or nD to a function f and all subfunctions until constant functions are reached. If the resulting expressions are expanded, a two-level AND-EXOR form called PSDKRO results.

Example 2. Let $f(x_1, x_2, x_3) = x_1x_2 + x_3$. If f is decomposed using S, we get:

$$f_{x_1}^0 = x_3 \text{ and } f_{x_1}^1 = x_2 + x_3$$

Then, decomposing $f_{x_1}^0$ using pD and $f_{x_1}^1$ using nD, we get:

$$(f_{x_1}^0)_{x_3}^0 = 0 \textit{ and } (f_{x_1}^0)_{x_3}^2 = 1$$
$$(f_{x_1}^1)_{x_2}^1 = 1 \textit{ and } (f_{x_1}^1)_{x_2}^2 = 1 \oplus x_3$$

Finally, again pD is applied for $(f_{x_1}^1)_{x_2}^2$:

$$((f_{x_1}^1)_{x_2}^2)_{x_3}^0 = 1 \textit{ and } ((f_{x_1}^1)_{x_2}^2)_{x_3}^2 = 1$$

Thus, by expanding the respective expressions, the following PSDKRO description for f results:

$$f = \overline{x}_1x_3 \oplus x_1 \oplus x_1\overline{x}_2 \oplus x_1\overline{x}_2x_3$$

3 ESOP-Based Synthesis

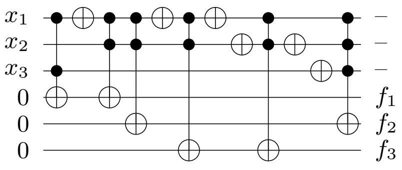
In this work, evolutionary algorithms are applied in order to improve the ESOP-based synthesis method originally introduced in [6]. For a given function $f : \mathbb{B}^n \to \mathbb{B}^m$, this approach generates a circuit with $n + m$ lines, whereby the first n lines also work as primary inputs. The last m circuit lines are respectively initialized to a constant 0 and work as primary outputs. Having that, gates are selected such that the desired function is realized. This selection exploits the fact that a single product $x_{i_1}, \ldots x_{i_k}$ of an ESOP description directly corresponds to a Toffoli gate with control lines $C = \{x_{i_1}, \ldots x_{i_k}\}$. In case of negative literals, NOT gates (i.e. Toffoli gates with $C = \emptyset$) are applied to generate the appropriate values. Based on these ideas, a circuit realizing a function given as ESOP can be derived as illustrated in following example.

Example 3. Consider the function f to be synthesized as depicted in Fig. 2(a)[1]. The first product x_1x_3 affects f_1. Accordingly, a Toffoli gate with control

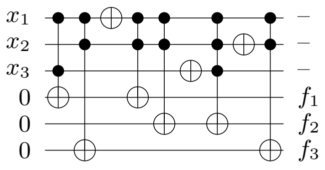
[1] The column on the left-hand side gives the respective products, where a "1" on the i^{th} position denotes a positive literal (i.e. x_i) and a "0" denotes a negative literal (i.e. $\overline{x}_i$), respectively. A "-" denotes that the respective variable is not included in the product. The right-hand side gives the respective primary output patterns.

x_1	x_2	x_3	f_1	f_2	f_3
1	-	1	1	0	0
0	1	-	1	1	0
1	1	-	0	0	1
0	0	-	0	0	1
0	1	0	0	1	0

(a) ESOP



(b) Basic approach



(c) W/ reordering

Fig. 2. ESOP-based synthesis

lines $C = \{x_1 x_3\}$ and a target line representing the primary output f_1 is added (see Fig. 2(b)). The next product $\overline{x}_1 x_2$ includes a negative literal. Thus, a NOT gate is needed at line x_1 to generate the appropriate value for the next mappings. Since $\overline{x}_1 x_2$ affects both, f_1 and f_2, two Toffoli gates with control lines $C = \{x_1 x_2\}$ are added next. Afterwards, a further NOT gate is applied to restore the value of x_1 (needed again by the third product). This procedure is continued until all products have been considered. The resulting circuit is shown in Fig. 2(b).

Note that thereby the order in which the respective products are traversed may have a slight impact on the resulting circuit cost. For example, the line x_1 in the circuit from Example 3 is unnecessarily often inverted. This can be avoided by treating the respective products in a different order as shown in Fig. 2(c). Here, the two product terms with positive literals only were considered first. Afterwards, the products including $\overline{x}_1$, $\overline{x}_1\overline{x}_3$, and, finally, $\overline{x}_1\overline{x}_2$ have been handled. This leads to a reduction in the number of NOT gates by 3. In the following, a reordering scheme as introduced in [6] is applied to generate the circuits.

Overall, having an ESOP description of the function f to be synthesized, a reversible circuit realizing f can easily be created using the reviewed approach. However, the quality of the resulting circuits strongly depends on the following properties of the given ESOP:

– The number of products (since for each product, a Toffoli gate is added to the circuit),
– the number of primary outputs affected by a product (since for each affected primary output, a Toffoli gate is added to the circuit), and
– the number of literals within a product (since for each literal, a control line needs to be added which causes additional cost as shown in Table 1).

These criteria contradict with the optimization goals applied in common Boolean optimization approaches (e.g. EXORCISM [19]), where usually only the number of product terms is reduced. In contrast, considering ESOP-based synthesis, a function description including more products might be better if instead the number of literals within these products is smaller. Then, although even more gates have to be added, these gates are of less cost. Determining a "good" ESOP representation trading-off these contradictory criteria is thereby a non-trivial task. The next section introduces an evolutionary algorithm addressing this problem.

4 EA-Based Optimization

In order to optimize a given ESOP description with respect to the criteria outlined in the previous section, the approach introduced in [7] is utilized. This approach optimizes PSDKRO descriptions – as mentioned above, a subclass of ESOPs enabling efficient optimization. In this section, we briefly review the essential parts of the algorithm and describe the extensions to address the new cost models.

4.1 General Flow

In [7], PSDKROs are optimized using *Reduced Ordered Binary Decision Diagrams (ROBDDs)* [2]. Having a BDD representing the function to be optimized, a depth-first traversal over all nodes is performed. Then, a PSDKRO is derived exploiting the fact that for each decomposition (i.e. for each S, pD, and nD) two out of three possible successors f_i^0, f_i^1, and f_i^2 are needed. That is, in order to generate the PSDKRO, for each node the costs of these three sub-functions are determined. Since ROBDDs are applied, f_i^0 and f_i^1 already are available. In case of f_i^2, the respective function representation is explicitly created. Having the respective costs, the two cheapest sub-functions are applied leading to the respective decomposition type for the PSDKRO.

Using this algorithm, a PSDKRO results which is optimal with respect to a given ordering of the ROBDD. However, modifying the variable ordering likely has an effect on the cost of the PSDKRO. Thus, determining a variable ordering leading to the best as possible PSDKRO representation remains as optimization task. Therefore, an evolutionary algorithm described as follows is applied.

4.2 Individual Representation

The ordering of the input variables in the expansion influences the cost of the PSDKRO. To obtain the minimum cost for all orderings, $n!$ different combinations have to be considered, where n denotes the number of input variables. That is, a permutation problem is considered. This can easily be encoded in EAs by means of vectors over n integers. Each vector represents a permutation, i.e. a valid ordering for the ROBDD, and works as individual in the EA. The population is a set of these elements.

4.3 Operators

In the proposed EA-approach, several procedures for recombination, mutation, and selection are applied. Due to page limitations, they are introduced in a brief manner. For a more detailed treatment, references to further readings are provided.

Crossover and Mutation. To create an offspring of a current population two crossover operators and three mutation operators are used alternately.

For recombination *Partially Matched Crossover* (PMX) [8] and *Edge Recombination Crossover* (ERX) [22] are applied equally. In our application, both operators create two children from two parents.

PMX: Choose two cut positions randomly. Exchange the parts between the cut positions in the parent individuals to create two children. Validate the new individuals in preserving the position and order of as many variables as possible.

ERX: Create an adjacency matrix which lists the neighbors of each variable in both parents. Beginning with an arbitrary variable, next the variable with the smallest neighbor set is chosen iteratively. Already chosen variables are removed from all neighbor sets.

For mutation three operators are used as follows:

SWAP: Randomly choose two positions of a parent and exchange the values of these positions.

NEIGHBOR: Select one position $i < n$ randomly and apply SWAP with positions i and $i + 1$.

INVERSION: Randomly select two positions i and j and invert all variables within i and j.

Selection. During the experimental evaluation, several selection procedures have been applied and the following turned out to be usually advantageous. As parent selection a deterministic tournament between q uniformly chosen individuals is carried out. The best individual is chosen as a parent used for recombination or mutation, respectively.

To determine the population for the next generation, PLUS-selection $(\mu + \lambda)$ is applied. Here, the best individuals of both, the current population μ and the offspring λ, are chosen equally. By this, the best individual never gets lost and a fast convergency is obtained.

4.4 Termination Criterions

The optimization process is aborted if no improvement is obtained for $20 * ln(n)$ generations or a maximum number of 500 generations, respectively. The default termination criterions are chosen based on experiments in a way that the EA provides a compromise between acceptable runtime and high quality results.

4.5 Parameter Settings

In general, all parameters and operators described above are parameterizable by the user. However, by default the size of the population is chosen two times larger than the number of primary inputs of the considered circuit. For the creation of the offspring, recombination is applied with a probability of 35% while mutation is used with a probability of 65%. By this, mutation also can be carried out on newly elements created by recombination.

4.6 Overall Algorithm and Fitness Function

At the beginning of an EA run, an initial population is generated randomly. Each of the individuals corresponds to a valid variable ordering of the ROBDD. In each generation an offspring of the same size of the parent population is created according to the operators described above. With respect to the fitness, the best individuals of both populations are chosen to be in the next generation. If a termination criterion is met, the best individual is returned.

As discussed above, the applied fitness function is thereby different. Instead of minimizing the number of products, further criteria need to be considered. To incorporate this into the EA, the respective cost functions from Table 1 is encoded and integrated in the selection procedure. More precisely, for each individual, the resulting PSDKRO (or ESOP, respectivley) description is traversed and the cost are added according to the quantum cost model or the transistor cost model, respectively. The resulting value is used as fitness for the considered individual.

Example 4. Consider the function *5xp1* from the the LGSynth benchmark library. This PSDKRO description originally has $p_{initial} = 48$ products which is also the optimal result using the original approach from [7]. However, the quantum cost are $qc_{initial} = 1081$. In contrast, if the proposed configuration is applied, a PSDKRO with an increase in the number of products to $p_{qcmin} = 50$ results. But, a circuit with quantum cost of only $qc_{qcmin} = 865$ can be derived. This shows that a decreasing number of products not coincidently means decreasing quantum cost or transistor cost, respectively.

5 Experimental Evaluation

The proposed approach has been implemented in C++ utilizing the EO library [10], the BDD package CUDD [18], and the RevKit toolkit [17]. As benchmarks, we used functions provided in the LGSynth package. All experiments have been carried out on an AMD 64-Bit Opteron 2,8 GHz with 32GB memory running linux.

The obtained results are summarized in Table 2. The first columns give the name of the respective benchmarks as well as the number of their primary inputs (denoted by *PIs*) and primary outputs (denoted by *POs*). Afterwards, the cost of the circuits generated from the initial PSDKRO representation (denoted by *Init. Cost*) and generated from the optimized PSDKRO representation (denoted by *Opt. Cost*) are provided. Furthermore, the resulting improvement (given in percent and denoted by *Impr.*) as well as the needed run-time (given in CPU seconds and denoted by *Time*) is listed. We distinguish thereby between the optimization with respect to quantum cost and the optimization with respect to transistor cost.

As can be seen, exploiting evolutionary algorithms significantly helps to reduce the cost of reversible circuits. For the majority of the benchmarks, double-digit improvement rates are achieved in less than an hour – in many cases just a couple of minutes is needed. If transistor cost is considered, the reductions are somewhat smaller. This was expected as this cost model is linear in comparison to the exponential quantum cost model (see Table 1). In the best case, quantum cost (transistor cost) can be reduced by 44.7% (37.4%) in less than 10 minutes (20 minutes).

Table 2. Experimental evaluation

Benchmark			Quantum Cost				Transistor Cost			
Name	PIs	POs	Init. Cost	Opt. Cost	Impr %	Time *s*	Init. Cost	Opt. Cost	Impr %	Time *s*
5xp1	7	10	1181	865	26.8	177.0	1424	1080	24.2	205.6
rd84	8	4	2072	2062	0.5	133.3	2528	2528	0.0	130.2
sym9	9	1	16535	16487	0.3	42.8	5088	5088	0.0	40.7
sym10	10	1	37057	35227	4.9	45.7	8408	7984	5.0	45.4
add6	12	7	5112	5084	5.5	370.1	5232	5232	0.0	348.6
alu2	10	6	5958	4476	24.9	188.2	4824	3960	17.9	165.0
alu4	14	8	79311	43850	44.7	594.1	56752	36784	35.2	494.6
apex4	9	19	59175	50680	14.4	315.3	54400	48552	10.8	338.3
b9	41	21	4237	3800	10.3	1331.2	4040	3632	10.1	1307.7
b12	15	9	1082	1049	3.0	417.8	1176	1112	5.4	412.1
con1	7	2	188	162	13.8	30.4	264	224	15.2	40.2
clip	9	5	5243	4484	14.5	151.8	4472	3808	14.8	164.4
duke2	22	29	11360	10456	8.0	1849.3	10016	9248	7.7	1846.8
log8mod	8	5	1118	941	15.8	102.3	1312	1160	11.6	101.6
misex1	8	7	475	466	1.9	190.0	608	608	0.0	185.0
misex3	14	14	82914	67206	18.9	940.8	72528	58464	19.4	890.5
misex3c	14	14	100481	85330	15.1	1016.6	88144	74544	19.4	850.0
sao2	10	4	6005	5147	14.3	141.6	3200	2704	15.5	154.4
spla	16	46	50399	49419	1.9	2498.5	42424	41672	1.8	2392.4
sqrt8	8	4	605	461	23.8	108.1	672	512	23.8	98.7
squar5	5	8	292	251	14.0	18.7	488	448	8.2	17.0
t481	16	1	275	237	13.8	56.1	352	320	9.1	55.2
table3	14	14	46727	35807	23.4	825.7	40208	30800	23.4	843.2
table5	17	15	54729	34254	37.4	1253.0	45408	28440	37.4	1147.7
ttt2	24	21	2540	2445	3.7	1216.2	2720	2584	5.0	1181.4
vg2	25	8	22918	18417	19.6	564.2	18280	14432	21.1	566.0

6 Conclusions

In this paper, an evolutionary algorithm is applied in order to improve ESOP-based synthesis of reversible circuits. By this, PSDKROs are considered which are a subclass of ESOPs. Reversible circuits received significant attention in the past – not least because of the promising applications in the domain of low-power design or quantum computation. ESOP-based synthesis is an efficient method for synthesis of this kind of circuits. Applying the proposed approach, the results obtained by this method can be improved significantly – in most of the cases by double-digit percentage points.

Acknowledgment

This work was supported by the German Research Foundation (DFG) (DR 287/20-1).

References

1. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. 17(6), 525–532 (1973)
2. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Comp. 35(8), 677–691 (1986)
3. Cuykendall, R., Andersen, D.R.: Reversible optical computing circuits. Optics Letters 12(7), 542–544 (1987)

4. Davio, M., Deschamps, J., Thayse, A.: Discrete and Switching Functions. McGraw-Hill, New York (1978)
5. Desoete, B., Vos, A.D.: A reversible carry-look-ahead adder using control gates. INTEGRATION, the VLSI Jour. 33(1-2), 89–104 (2002)
6. Fazel, K., Thornton, M.A., Rice, J.E.: ESOP-based Toffoli gate cascade generation. In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 206–209 (2007)
7. Finder, A., Drechsler, R.: An evolutionary algorithm for optimization of pseudo kronecker expressions. In: Int'l Symp. on Multi-Valued Logic, pp. 150–155 (2010)
8. Goldberg, D., Lingle, R.: Alleles, loci, and the traveling salesman problem. In: Int'l Conference on Genetic Algorithms, pp. 154–159 (1985)
9. Gupta, P., Agrawal, A., Jha, N.K.: An algorithm for synthesis of reversible logic circuits. IEEE Trans. on CAD 25(11), 2317–2330 (2006)
10. Keijzer, M., Merelo, J.J., Romero, G., Schoenauer, M.: Evolving objects: a general purpose evolutionary computation library. In: Int'l Conference in Evolutionary Algorithms. pp. 231–244 (2001), the EO library is available at `eodev.sourceforge.net`
11. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. 5, 183 (1961)
12. Merkle, R.C.: Reversible electronic logic using switches. Nanotechnology 4, 21–40 (1993)
13. Miller, D.M., Maslov, D., Dueck, G.W.: A transformation based algorithm for reversible logic synthesis. In: Design Automation Conference, pp. 318–323 (2003)
14. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge Univ. Press, Cambridge (2000)
15. Reed, I.: A class of multiple-error-correcting codes and their decoding scheme. IRE Trans. on Inf. Theory 3, 6–12 (1954)
16. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. on CAD 22(6), 710–722 (2003)
17. Soeken, M., Frehse, S., Wille, R., Drechsler, R.: RevKit: a toolkit for reversible circuit design. In: Workshop on Reversible Computation (2010), RevKit is available at `http://www.revkit.org`
18. Somenzi, F.: CUDD: CU Decision Diagram Package Release 2.3.1. University of Colorado at Boulder (2001), CUDD is available at `www.vlsi.colorado.edu/~fabio/CUDD/`
19. Song, N., Perkowski, M.: Minimization of exclusive sum of products expressions for multi-output multiple-valued input, incompletely specified functions. IEEE Trans. on CAD 15(4), 385–395 (1996)
20. Thomson, M.K., Glück, R.: Optimized reversible binary-coded decimal adders. J. of Systems Architecture 54, 697–706 (2008)
21. Toffoli, T.: Reversible computing. In: de Bakker, W., van Leeuwen, J. (eds.) Automata, Languages and Programming, p. 632. Springer, Heidelberg (1980), technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.
22. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesman: The genetic edge recombination operator. In: Int'l Conference on Genetic Algorithms, pp. 133–140 (1989)
23. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: Design Automation Conference, pp. 270–275 (2009)
24. Zhirnov, V.V., Cavin, R.K., Hutchby, J.A., Bourianoff, G.I.: Limits to binary logic switch scaling – a gedanken model. Proc. of the IEEE 91(11), 1934–1939 (2003)

Evolution of Test Programs Exploiting a FSM Processor Model

Ernesto Sanchez, Giovanni Squillero, and Alberto Tonda

Dipartimento di Automatica e Informatica
Politecnico di Torino, Italy
{ernesto.sanchez,giovanni.squillero,alberto.tonda}@polito.it

Abstract. Microprocessor testing is becoming a challenging task, due to the increasing complexity of modern architectures. Nowadays, most architectures are tackled with a combination of scan chains and Software-Based Self-Test (SBST) methodologies. Among SBST techniques, evolutionary feedback-based ones prove effective in microprocessor testing: their main disadvantage, however, is the considerable time required to generate suitable test programs.

A novel evolutionary-based approach, able to appreciably reduce the generation time, is presented. The proposed method exploits a high-level representation of the architecture under test and a dynamically built Finite State Machine (FSM) model to assess fault coverage without resorting to time-expensive simulations on low-level models. Experimental results, performed on an OpenRISC processor, show that the resulting test obtains a nearly complete fault coverage against the targeted fault model.

Keywords: SBST microprocessor testing.

1 Introduction

In the last years, the market demand for a higher computational performance in embedded devices has been continuously increasing for a wide range of application areas, from entertainment (smart phones, portable game consoles), to professional equipment (palmtops, digital cameras), to control systems in various fields (automotive, industry, telecommunications). The largest part of today's Systems-on-Chip (SoCs) includes at least one processor core. Companies have been pushing design houses and semiconductor producers to increase microprocessor speed and computational power while reducing costs and power consumption. The performance of processor and microprocessor cores has impressively increased due to technological and architectural aspects. Microprocessor cores are following the same trend of high-end microprocessors and quite complex units may be easily found in modern SoCs.

Technology advancements impose new challenges to microprocessor testing: as device geometries shrink, deep-submicron delay defects are becoming more prominent [5], thereby increasing the need for at-speed tests; as core operating frequency and speed of I/O interfaces rise, more expensive external test equipment is required.

© Springer-Verlag Berlin Heidelberg 2011

The increasing size and complexity of microprocessor architectures directly reflects in more demanding test generation and application strategies. Modern designs contain intricate architectures that increase test complexity. Indeed, pipelined and superscalar designs demonstrated to be random pattern resistant [2]. The use of hardware-based approaches, such as scan chains and BIST, even though consolidated in industry for integrated digital circuits, has proven to be often inadequate, since these techniques introduce excessive area overhead [1], require extreme power dissipation during the test application [13], and are often ineffective when testing delay-related faults [12].

As a consequence, academy is looking for novel paradigms to respond to the new testing issues: one promising alternative to hardware-based approaches is to exploit the processor to execute carefully crafted test programs. The goal of these test programs is to uncover possible design or production flaws in the processor. This technique, called Software-Based Self-Test (SBST), has been already used in different problems with positive results.

In this paper, we propose a SBST simulation-based framework for the generation of post-production test programs for pipelined processors. The main novelty of the proposed approach is its ability to efficiently generate test programs, exploiting a high level description of the processor under test, while the evolution of the generation is driven by the transition coverage of a FSM created during the evolution process itself.

The rest of the paper is organized as follows: section 2 provides background on SBST and SBST-related evolutionary computation. Section 3 describes the proposed methodology. Section 4 outlines the case study and reports the experimental results, and section 5 drafts some conclusions of our work.

2 Background

2.1 SBST

SBST techniques have many advantages over other testing methodologies, thanks to their features: the testing procedure can be conducted with very limited area overhead, if any; the average power dissipation is comparable with the one observable in mission mode; the possibility of damages due to excessive switching activity, non-negligible in other methods, is virtually eliminated; test programs can be run at the maximum system speed, thus allowing testing of a larger set of defects, including delay-related ones; the approach is applicable even when the structure of a module is not known or cannot be modified.

SBST approaches proposed in literature do not necessarily aim to substitute other established testing approaches (e.g., scan chains or BIST) but rather to supplement them by adding more test quality at a low cost. The objective is to create a test program able to run on the target microprocessor and test its modules, satisfying the target fault coverage requirements. Achieving this test quality requires a proper test program generation phase, which is the main focus of most SBST approaches in recent literature. The quality of a test program is measured by its coverage of the design errors or production defects, its code size, and time required for its execution.

The available approaches for test program generation can be classified according to the processor representation that is employed in the flow. High-level representations of the processor Instruction Set Architecture (ISA) or state transition graphs are convenient for limiting the complexity of the architecture analysis, and provide direct correlation with specific instruction sequences, but cannot guarantee the detection of structural faults. Lower-level representations, such as RT and gate-level netlists, describe in greater detail the target device and allow to focus on structural fault models, but involve additional computational effort.

A survey on some of the most important techniques developed for test program generation is presented in [9]. Due to modern microprocessors' complex architectures, automatic test program generation is a challenging task: considering different architectural paradigms, from pipelined to multithreaded processors, the search space to be explored is even larger than that of classic processors. Thus, it becomes crucial to devise methods able to automate as much as possible the generation process, reducing the need for skilled (and expensive) human intervention, and guaranteeing an unbiased coverage of corner cases.

Generation techniques can be classified in two main groups: *formal* and *simulation-based*. Formal methodologies exploit mathematical techniques to prove specific properties, such as the absence of deadlocks or the equivalence between two descriptions. Such a proof implicitly considers all possible inputs and all accessible states of the circuit. Differently, simulation-based techniques rely on the simulation of a set of stimuli to unearth misbehaviors in the device under test. A simulation-based approach may therefore be able to demonstrate the presence of a bug, but will never be able to prove its absence: however, the user may assume that the bug does not exist with level of confidence related to the quality of the simulated test set. The generation of a qualifying test set is the key problem with simulation-based techniques. Different methodologies may be used to add contents to such test sets, ranging from deterministic to pseudo-random.

Theoretically, formal techniques are able to guarantee their results, while simulation-based approaches can never reach complete confidence. However, the former require considerable computational power, and therefore may not be able to provide results for a complex design. Moreover, formal methodologies are routinely applied to simplified models, or used with simplified boundaries conditions. Thus, the model used could contain some differences with respect to the original design, introducing a certain amount of uncertainty in the process [8].

Nowadays, simulation-based techniques dominate the test generation arena for microprocessors, with formal methods bounded to very specific components in the earliest stages of the design. In most of the cases, simulation-based techniques exploit *feedback* to iteratively improve a test set in order to maximize a given target measure. Nevertheless, simulation of low-level descriptions could require enormous efforts in terms of computational time, memory occupation and hardware.

The main drawback of feedback-based simulation methods, is the long elaboration time required during the evolution. When dealing with a complete processor core, for example in [10], the generation time increases when low abstraction descriptions are used as part of the evolution: the growth of computation times is mainly due to the inclusion of *fault simulation* in the process. For every possible fault of the design, the original circuit is modified including the considered fault; then, a complete simulation is performed in order to understand whether the fault changes the circuit outputs. Even though lots of efforts are spent on improving this process [6], several minutes are still required to perform a fault simulation on a processor core with about 20k faults.

2.2 EAs on SBST

Several approaches that face test program generation by exploiting an automated methodology have been presented in recent years: in [11] a tool named VERTIS, able to generate both test and verification programs based on the processor's instruction set architecture only, is proposed. VERTIS generates many different instruction sequences for every possible instruction being tested, thus leading to very large test programs. The test program generated for the GL85 processor following this approach is compared with the patterns generated by two Automatic Test Pattern Generator (ATPG) tools: the test program achieves a 90.20% stuck-at fault coverage, much higher than the fault coverage of the ATPG tools, proving the efficacy of SBST for the first time. The VERTIS tool works with either pseudo-random instruction sequences and random data, or with test instruction sequences and heuristics to assign values to instruction operands specified by the user in order to achieve good results. In more complex processors, devising such heuristics is obviously a non-trivial task.

In [7], an automated functional self-test method, called *Functional Random Instruction Testing at Speed* (FRITS), is presented. FRITS is based on the generation of random instruction sequences with pseudorandom data. The authors determine the basic requirements for the application of a cache-resident approach: the processor must incorporate a cache load mechanism for the test program downloading and the loaded test program must not produce either cache misses or bus cycles. The authors report some results on an Intel Pentium® 4 processor: test programs automatically generated by the FRITS tool achieve 70% stuck-at fault cover-age for the entire chip, and when these programs are enhanced with manually generated tests, the fault coverage increases by 5%.

Differently from the previously described functional methods, in [3] the authors propose a two-steps methodology based on evolutionary algorithms: firstly a set of macros encrypting processor instructions is created, and in a second step an evolutionary optimizer is exploited to select macros and data values to conform the test program. The proposed approach is evaluated on a synthesized version of an 8051 microprocessor, achieving about 86% fault coverage. Later, in [2], a new version of the proposed approach is presented. The authors exploit a simulation-based method that makes use of a feedback evaluation to improve the quality of test programs: the approach is based on an evolutionary algorithm and it is capable of evolving small test programs that capture target corner cases for design validation purposes. The effectiveness of the approach is demonstrated by comparing it with a pure instruction randomizer, on a RTL description of the LEON2 processor. With respect to the purely random method, the proposed approach is able to seize three additional intricate corner cases while saturating the available addressed code coverage metrics. The developed validation programs are more effective and smaller in code size.

3 Proposed Approach

The previously described test generation cases show that evolutionary algorithms can effectively face real-world problems. However, when exploiting a low-level description of the processor under evaluation, simulation-based approaches require huge elaboration times.

We propose a methodology able to exploit a high-level description of a pipelined processor core in the generation process: the required generation time is thus reduced with respect to techniques that use a low-level description during the generation phase, such as the gate-level netlist, as reported in [10]. In the proposed approach, it must be noticed that the processor netlist is only used at the end of the generation process to assess the methodology results, performing a complete fault simulation.

The generation process is supported by the on-time automated generation of a FSM that models the excited parts of the processor core and drives the evolution process by indicating the unreached components on the processor core. In addition, we consider also high-level coverage metrics to improve the evolution.

It is possible to simply define a pipelined microprocessor as the interleaving of sequential elements (data, state and control registers), and combinational logic blocks. The inputs of the internal combinatory logic blocks are dependent on the instruction sequence that is executed by the processor and on the data that are processed.

One way to model a microprocessor is to represent it with a FSM. Coverage of all the possible transitions in the machine ensures thoroughly exercising the system functions. Additionally, the use of the FSM transition coverage has the additional advantage that it explicitly shows the interactions between different pipeline stages. Thus, we define the *state word* of a pipelined processor FSM model as the union of all logic values present in the sequential elements of the pipeline, excluding only the values strictly related to the data path. Consequently, the FSM transits to a new state at every clock cycle, because at least one bit in the state word is changed due to the whole interaction of the processor pipeline.

Figure 1 shows the proposed framework. The evolutionary core, called *μGP^3* [15], is able to generate syntactically correct assembly programs by acquiring information about the processor under evaluation from an user-defined file called *Constraint Library*. When the process starts, the evolutionary core generates an initial set of random programs, or individuals, exploiting the information provided by the library of constraint. Then, these individuals are cultivated following the Darwinian concepts of natural evolution. Every test program is evaluated resorting to external tools that simulate the high level description of the processor core, resorting to a logic simulator at RTL, and generate a set of high-level measures. Contemporary, during the logic simulation, the FSM status is captured at every clock cycle, and for every evaluated test program the visited states and the traveled transitions are reported back to the evolutionary core as part of the evaluation of the goodness of an individual, called *fitness value*. The interaction between the different elements composing the fitness value guarantees good quality regarding the fault coverage against a specific fault model at gate level. Fitness values gathered during the logic simulation, for example code coverage metrics such as *Statement coverage* (SC), *Branch coverage* (BC), *Condition coverage* (CC), *Expression coverage* (EC), *Toggle coverage* (TC), are suitable for guiding the evolution of test programs. Simultaneously, maximizing the number of traversed transitions of the FSM model, assures a better result at gate level.

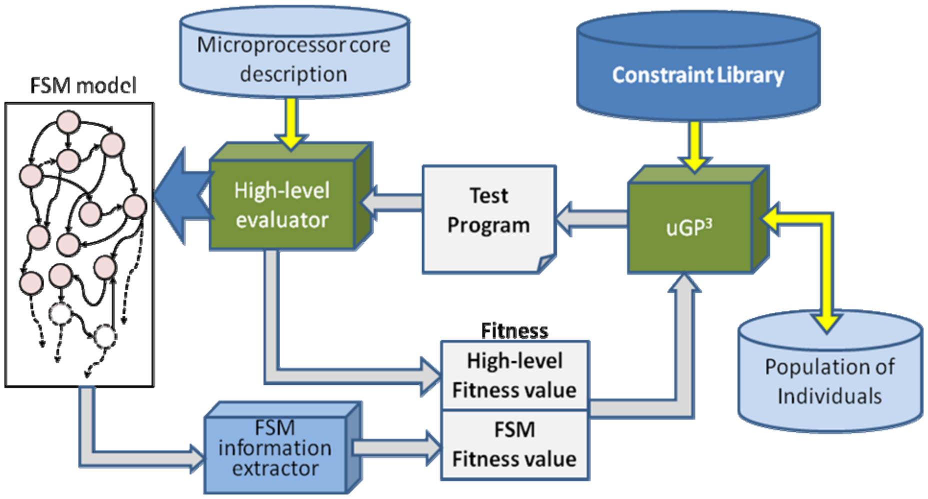


Fig. 1. Test generation framework.

Only the best individual is fault simulated in order to assess its fault coverage properties, reducing generation times. In the following paragraphs, we briefly describe in more detail the most significant elements present in the described framework.

3.1 µGP^3

µGP^3 represent individuals, in this case candidate test programs, as constrained tagged graphs; a tagged graph is a directed graph every element of which may own one or more tags, and that in addition has to respect a set of constraints. A tag is a name-value pair used to add semantic information to graphs, augmenting the nodes with a number of parameters, and also to uniquely identify each element during the evolution. Graphs are initially generated in a random fashion; subsequently, they may be modified by genetic operators, such as the classical mutation and recombination. The genotype of an individual is described by one or more constrained tagged graphs.

The purpose of the constraints is to limit the possible productions of the evolutionary tool, also providing them with semantic value. The constraints are provided through a user-defined library that supplies the genotype-phenotype mapping for the generated individuals, describes their possible structure and defines which values the existing parameters (if any) can assume. To increase the generality of the tool, constraint definition is left to the user.

In this specific case the constraints define three distinct sections in an individual: a program configuration part, a program execution part and a data part or stimuli set. The first two are composed of assembly code, the third is written as part of a VHDL testbench. Though syntactically different, the three parts are interdependent in order to obtain good solutions.

Individual fitness values are computed by means of one or more external evaluator tools. The fitness of an individual is represented by a sequence of floating point numbers optionally followed by a comment string. This is currently used in a prioritized fashion: one fitness A is considered greater than another fitness B if the n-th component of A is greater than the n-th component of B and all previous components (if any) are equal; if all components are equal then the two fitnesses are considered equal.

The evolutionary tool is currently configured to cultivate all individuals in a single panmictic population. The population is ordered by fitness. Choice of the individuals for reproduction is performed by means of a tournament selection; the tournament size τ is also endogenous. The population size μ is set at the beginning of a run, and the tool employs a variation on the plus (μ+λ) strategy: a configurable number λ of genetic operators are applied on the population. All new unique individuals are then evaluated, and the population resulting from the union of old and new individuals is sorted by decreasing fitness. Finally, only the first μ individuals are kept.

The possible termination conditions for the evolutionary run are: a target fitness value is achieved by the best individual; no fitness increase is registered for a predefined number of generations; a maximum number of generations is reached.

3.2 FSM Extractor

The proposed methodology is based on modeling the entire processor core as a FSM which is dynamically constructed during the test generation process. Thus, differently from other approaches, the FSM extraction is fully automated, and demands minimum human effort: the approach only requires the designer to identify the memory elements of the pipeline registers in the RTL processor description that will determine state characteristics of the FSM. The key point behind the FSM extractor is to guide the evolution trough a high-level model of the processor core that summarizes the capacity of excitation of the considered test program. The FSM information extractor receives from the external evaluator (e.g., a logic simulator) the activity of the pipeline registers of the processor core at every clock cycle, then, it computes for every clock cycle the processor state word and extracts the visited states and the traversed transitions.

Given the dynamic nature of the FSM construction, it is not possible to assume as known the maximum number of reachable states, not to mention the possible transitions. For this reason, it is impossible to determine the transition coverage with respect to the entire FSM.

The implemented evaluator, that includes the logic simulator and the FSM information extractor, collects the output of the simulation and dynamically explores the FSM; it assesses the quality of test program considering the transition coverage on the FSM and the code coverage metrics. The fitness fed back to the evolutionary tool is composed of many parts: the FSM transition coverage followed by all other high-level metrics (SC, BC, CC, EC, TC).

Let us consider the mechanisms related to hazard detection and forwarding activation in a pipelined processor: in order to thoroughly test them, it requires to stimulate the processor core with special sequences of strongly dependent instructions able to activate and propagate possible faults on these pipelined mechanisms. Facing this problem by hand requires a very good knowledge about the processor core to carefully craft a sequence of instructions able to actually excite the mentioned pipelined elements. Additionally, this process may involve a huge quantity of time.

On the other hand, state-of-the-art test programs usually do not target such pipeline mechanisms, since their main concern is exciting a targeted functional unit through carefully selected values, and not to activate the different forwarding paths and other mechanisms devoted to handle data dependency between instructions [4].

As a matter of fact, it is possible to state that a feedback based approach able to collect information about the interaction of the different instructions in a pipelined processor as the one described before, allows the evolution of sequences of dependent instructions that excite the mentioned pipeline mechanisms.

4 Case Study and Experimental Results

The effectiveness of the EA-based proposed methodology has been experimentally evaluated on a benchmark SoC that contains the OpenRISC processor core and some peripheral cores, such as the VGA interface, PS/2 interface, Audio interface, UART, Ethernet and JTAG Debug interface. The SoC uses a 32 bit WISHBONE bus rev. B for the communication between the cores. The operating frequency of the SoC is 150 MHz. The implemented SoC is based on a version publicly available at [14].

The OpenRISC processor is a 32 bit scalar RISC architecture with Harvard microarchitecture, 5 stages integer pipeline and virtual memory support. It includes supplementary functionalities, such as programmable interrupt controller, power management unit and high-resolution tick timer facility. The processor implements a 8Kbyte data cache and a 8Kbyte instruction cache 1-way direct mapped; the instruction cache is separated from the data cache because of the specifics of the Harvard microarchitecture.

In our experiments we decide to tackle specifically the processor integer unit (IU) that includes the whole processor pipeline. This unit is particularly complex and important in pipelined processors, since it is in charge of handling the flow of instructions elaborated in the processor core.

The pipelined processor is described by eight verilog files, counting about 4,500 lines of code. Table 1 describes some figures that are used to compute RTL code coverage and toggle metrics. Additionally, the final line shows the number of stuck-at faults (*S@ faults*) present in the synthesized version of the targeted module.

The state word is defined as the union of all memory elements composing the processor pipeline, excluding only the registers that contain data elements. Data registers are excluded because we are mainly interested in the control part of the pipeline, and not in the data path.

Table 1. IU information facts

OR1200 IU	
Lines	4,546
Statements	466
Branches	443
Condition	53
Expression	123
Toggle	3,184
S@ faults	13,248

Thus, considering the registers available in every stage of the processor pipeline, a state word contained 237 bits is saved at every clock cycle during the logic simulation of a test program allowing us to dynamically extract the processor FSM.

In order to monitor the elements contained in the state word of the pipeline at every clock cycle, we implemented a *Programming Language Interface* module, called *PLI*, that captures the information required during the logic simulation of the RTL processor. The PLI module is implemented in C language, counting about 200 lines of code. The module is compiled together with the RTL description of the processor core, exploiting a verilog wrapper.

Once a test program is simulated, a script *perl* extracts the information regarding the number of visited states as well as the number of traversed transitions obtained by the considered program. This information is collected together to the high-level coverage metrics provided by the logic simulator and the complete set of values is fed back to the evolutionary engine in the form of fitness value of the test program.

The configuration files for the evolutionary optimizer are prepared in XML and count about 1,000 lines of code. Finally, additional *perl* scripts are devised to close the generation loop.

A complete experiment targeting the OR1200 pipeline requires about 5 days. At the end of the experiment, an individual counting 3,994 assembly lines that almost saturate the high level metrics is created; the same individual obtains about 92% fault coverage against the targeted fault model.

Compared to manual approaches reported in [4], that achieve about 90% fault coverage in the considered module, the results obtained in this paper improve the fault coverage by about 2%, and can be thus considered promising.

5 Conclusions

Tackling microprocessor testing with evolutionary algorithms proved effective in many works in literature, but this methodologies share a common disadvantage: the time needed to evolve a suitable test program is considerable.

In order to solve this problem, a novel evolutionary-based test approach is proposed. The approach exploits a high-level description of the device under test, along with a dynamically built FSM model, to esteem the fault coverage of the candidate test programs. Thus, a reliable evaluation of the goodness of the programs is obtained without resorting to time-expensive simulations on low-level models.

The proposed framework is assessed on a OpenRISC processor. Experimental results show a total of 92% fault coverage against the targeted fault model.

Acknowledgements. We would like to thank Danilo Ravotto and Xi Meng for technical assistance during experimental evaluation.

References

[1] Bushard, L., Chelstrom, N., Ferguson, S., Keller, B.: DFT of the Cell Processor and its Impact on EDA Test Software. In: IEEE Asian Test Symposium, pp. 369–374 (2006)

[2] Corno, F., Sanchez, E., Sonza Reorda, M., Squillero, G.: Automatic Test Program Generation – a Case Study. IEEE Design & Test of Computers 21(2), 102–109 (2004)

[3] Corno, F., Sonza Reorda, M., Squillero, G., Violante, M.: On the Test of Microprocessor IP Cores. In: DATE, pp. 209–213 (2001)

[4] Gizopoulos, D., Psarakis, M., Hatzimihail, M., Maniatakos, M., Paschalis, A., Raghunathan, A., Ravi, S.: Systematic Software-Based Self-Test for pipelined processors. IEEE Transactions on Very Large Scale Integration (VLSI) 16(11), 1441–1453 (2008)

[5] Mak, T.M., Krstic, A., Cheng, K.-T., Wang, L.-C.: New challenges in delay testing of nanometer, multigigahertz designs. IEEE Design & Test of Computers 21(3), 241–248 (2004)

[6] May, G., Spanos, C.: Fundamentals of Semiconductor Manufacturing and Process Control, p. 428. Wiley-IEEE Press publisher (2006) ISBN: 9780471790280

[7] Parvathala, P., Maneparambil, K., Lindsay, W.: FRITS – A Microprocessor Functional BIST Method. In: IEEE Intl. Test Conf., pp. 590–598 (2002)

[8] Pradhan, D.K., Harris, I.G.: Practical Design Verification. Cambridge University Press, Cambridge (2009) ISBN: 9780521859721

[9] Psarakis, M., Gizopoulos, D., Sanchez, E., Sonza Reorda, M.: Microprocessor Software-Based Self-Testing. IEEE Design & Test of Computers 27(3), 4–19 (2010)

[10] Sanchez, E., Sonza Reorda, M., Squillero, G.: Test Program Generation From High-level Microprocessor Descriptions. In: Test and Validation of Hardware/Software Systems Starting from System-level Descriptions, 179 p. Springer, Heidelberg (2005) ISBN: 1-85233-899-7, pp. 83-106

[11] Shen, J., Abraham, J.: Native mode functional test generation for processors with applications to self-test and design validation. In: Proceedings IEEE Intl. Test Conf., pp. 990–999 (1998)

[12] Speek, H., Kerchoff, H.G., Sachdev, M., Shashaani, M.: Bridging the Testing Speed Gap: Design for Delay Testability. In: IEEE European Test Workshop, pp. 3–8 (2000)

[13] Wang, S., Gupta, S.K.: ATPG for heat dissipation minimization during scan testing. In: ACM IEEE Design Automation Conference, pp. 614–619 (1997)

[14] `http://www.opencores.org/`

[15] `http://ugp3.sourceforge.net/`

Enhanced Reverse Engineering Using Genetic-Algorithms-Based Experimental Parallel Workflow for Optimum Design

Damir Vučina and Igor Pehnec

FESB, Faculty of Electrical Engineering,
Mechanical Engineering and Naval Architecture, University of Split
R. Boskovica bb, 21000 Split, Croatia
vucina@fesb.hr

Abstract. Shape optimization is numerically very intensive due to multidisciplinary objectives and constraints, many shape variables, non linear models, geometric infeasibility of candidate designs, etc. It involves participation of numerical optimizers, computer- aided geometric modelers and subject-related simulators as well as their coupling at the process- and data levels. This paper develops a simple experimental workflow which employs existing commercial software for computer-aided design, finite element analysis and evolutionary optimization modules. It sets up parallel execution of multiple simulators to reduce the execution time, which is implemented inexpensively by means of a self-made *.net*- based cluster. Shape optimization is introduced in the generic context of 'enhanced' reverse engineering with optimization whereby the initial solution can be obtained by 3D optical scanning and parameterization of an existing solution.

1 Introduction

There are very strict requirements in contemporary product development: the resulting products need to perform competitively according to given excellence benchmarks, they should be modest in terms of investment and operational costs, their respective time-to-market should be very short. As these largely conflicting requirements are very difficult to accomplish, some of the new conceptual approaches that have recently reached satisfactory levels of maturity seem complementary and potentially effective as modules in the overall product development process, Fig. 1.

Shape optimization is the key element in the design of a product, as it defines the shape that best provides for the required functionality within the given environment. In some cases it follows the phase of topology optimization, other approaches combine the two phases. The corresponding material distribution problem is obtained, where a given amount of material is to be distributed in the domain such that the structural response is optimized. The material distribution problem is a two-field problem of finding the optimal distributions over the design domain of the material distribution field (or material properties field) and the corresponding displacement field (or generally structural response field). It can be solved by 0/1 integer

© Springer-Verlag Berlin Heidelberg 2011

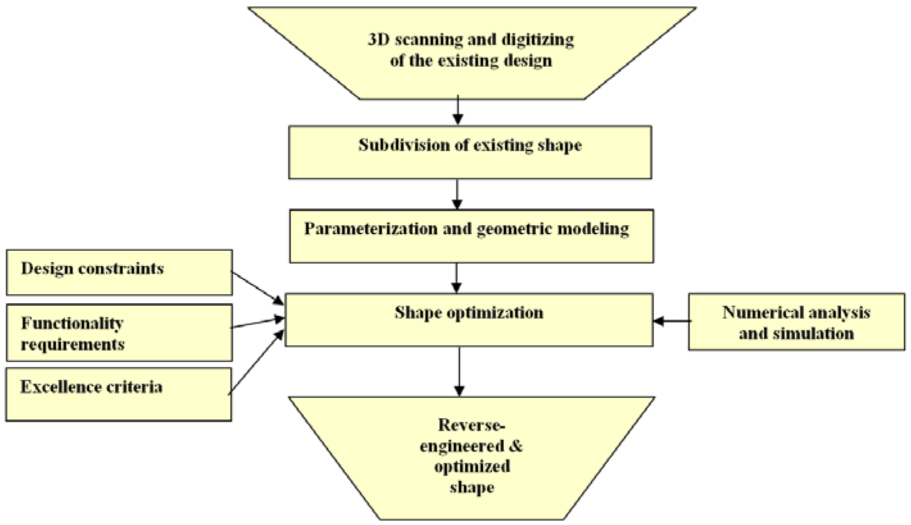

3D scanning and digitizing of the existing design
Subdivision of existing shape
Parameterization and geometric modeling
Design constraints
Functionality requirements
Excellence criteria
Shape optimization
Numerical analysis and simulation
Reverse-engineered & optimized shape

Fig. 1. Reverse engineering with shape optimization

programming or relaxed by the SIMP method. Subsequently the boundaries are obtained by numerical fitting and smoothening. Other approaches model the boundaries of the domain (external and internal) rather than the material distribution within the domain and typically apply parametric curves or surfaces. In any case, shape synthesis by means of optimization is responsible for generating the geometry of the object which directly determines its functionality and performance. By linking the shape synthesis process with numerical simulators of the interaction with the environment, virtual prototyping is fully provided for.

Shape optimization numerically implements the search for the ‘best’ feasible solution based on the requested functionality and excellence criteria, given the mathematical models describing these elements, [1,2]. Such design synthesis should result in shapes that provide the requested functionalities and maximize the given excellence criteria under given conditions. This paper develops the elements and layout for a simple experimental computational workflow to carry out shape optimization by deploying existing software as part of the overall process. It includes geometric modeling applications (CAD), [3,4], evolutionary optimizers (such as GA, genetic algorithms [5,6]), and simulation programs to evaluate the performance of candidate designs (finite-element-analysis, FEA), [7-9]. The experimental workflow also provides for coordination of computational processes and data exchange amongst the individual programs.

The optimizer searches the design space steered by excellence criteria and design constraints which are evaluated by calling the respective simulators such as FEA. The design space is spanned by shape parameters (parametric curves or surfaces) as independent variables. The candidate designs being synthesized by the evolutionary optimizer are assigned to the simulators for the purpose of evaluation of objective functions and constraints.

The overall optimum design procedure is here developed according to Fig. 2,

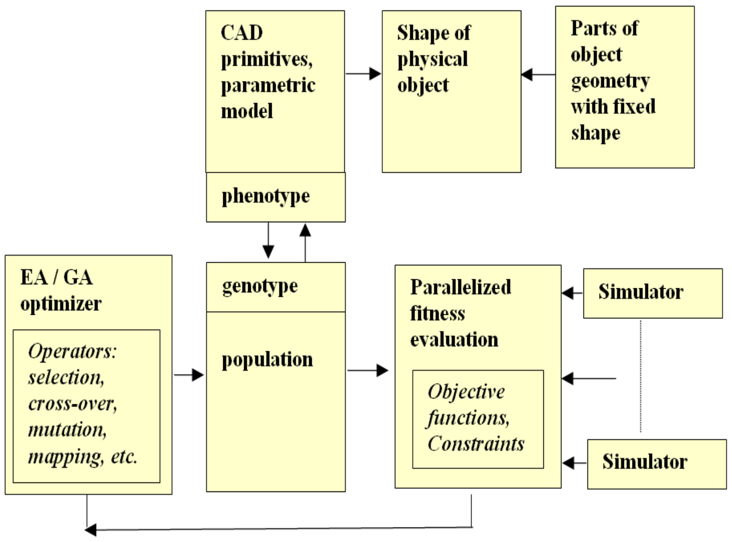

CAD primitives, parametric model
phenotype
Shape of physical object
Parts of object geometry with fixed shape
EA / GA optimizer
Operators: selection, cross-over, mutation, mapping, etc.
genotype
population
Parallelized fitness evaluation
Objective functions, Constraints
Simulator
Simulator

Fig. 2. EA / GA based shape optimization

The standard formulation of the optimum design problem reads: determine $\boldsymbol{x}$ ε R^n such that

$$\begin{aligned} &\min\{f(x)\} \\ &g_j(x) \le 0 \quad , j = 1, p \\ &h_j(x) = 0 \quad , j = 1, r \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ are the shape variables, $f(\boldsymbol{x})$ the objective function, and $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ the constraints.

2 Digitizing and Representation of Shape, Parameterization

Before the methodology is developed, the problem definition needs to be structured properly. This paper develops a generic computational workflow for shape optimization that can be set up using standard off-the-shelf programs (such as simulators) and in-house developed middleware (such as synchronization, parallelization, data-mining). The shape optimization will not start from scratch as existing objects will serve as initial solutions instead and make this process one of reverse engineering with optimization. These shapes can be scanned in 3D and parameterized by fitting mathematical surfaces, the control points of which also provide the set of shape variables. Subsequently, evolutionary algorithms are deployed to optimize such initial shape for best performance within a custom-developed cluster that implements parallelization for increased numerical efficiency.

Digitizing 3D shape based on optical methods, triangulation, and in particular stereo-photogrammetry, is discussed by many authors, for example [10]-[12]. Recently, there are also off-the-shelf systems available (eg. [13]) which provide the corresponding technology in high resolution and accuracy. Parameterizations of the resulting 3D point clouds typically by applying B-splines or NURBS ([3],[4]) are the subject of many papers related to corresponding best fitting procedures, for example [14]-[19].

The initial solution is obtained by 3D optical scanning of the geometry of the existing object. Fig. 3 shows the single-camera set-up with *c* denoting the camera and *w* the world coordinate systems respectively.

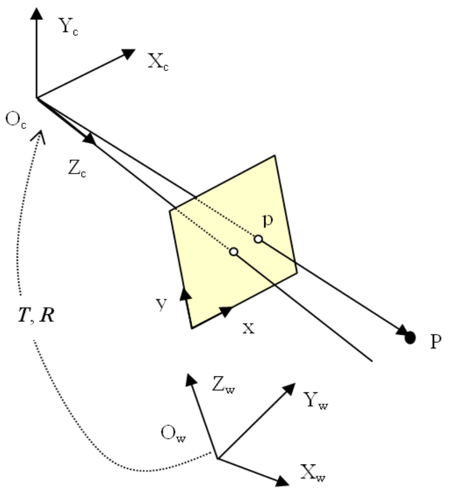


Fig. 3. Optical scanning for physical point ***P***: image point ***p***, camera (*c*) and world (*w*) coordinate systems, translation (*T*) and rotation (*R*), single camera view

If a stereo-camera set-up is used, two sets of image coordinates (*x*, *y*) for any point *P* are obtained, which make 3D image reconstruction possible. 3D reconstruction (2) recovers 3D Euclidean coordinates from stereo measurements (*l*, *r* denote the left and right cameras respectively) and can be done fully only if all extrinsic and intrinsic system parameters are known and calibrated.

$$(\mathbf{p}_l, \mathbf{p}_r) \Rightarrow \mathbf{P} \tag{2}$$

In our lab we use the ATOS 3D digitizing system, Fig. 4, where light patterns are projected onto the surface of the object to provide for spatial identification of a large number of points on the surface. The sequenced stripe patterns provide for time-based coding of position.

The 3D scanning step provides dense point clouds as output. The overall geometry of the object is partitioned into segments, some of which will be shape-optimized. In

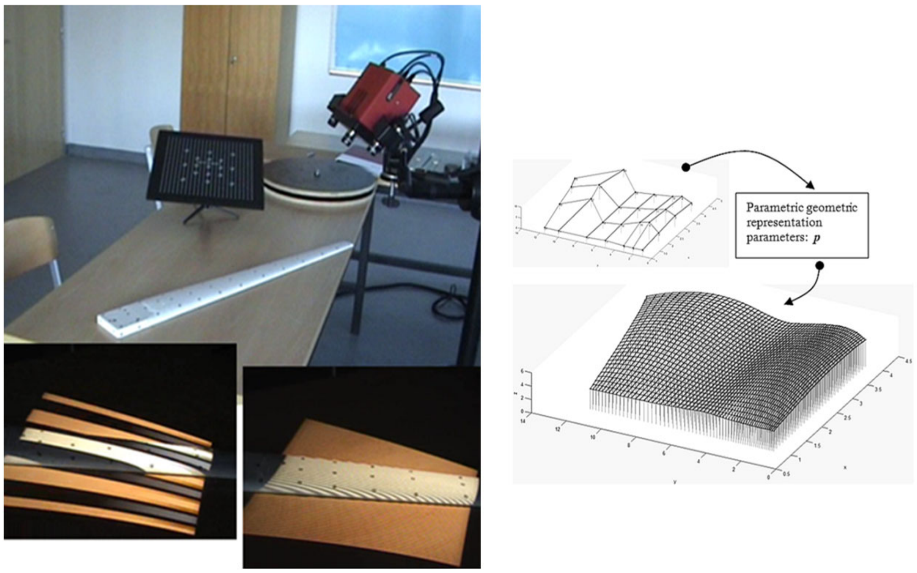


Fig. 4. 3D digitizer with structured light and (conceptually) parameterization of portion of shape by best-fitting a parametric surface to resulting points cloud (after polygonization)

order to do this, the corresponding point clouds will be parameterized. This implies that generic geometric modeling primitives will be selected to represent such segments and subsequently those generic primitives will be particularized for the respective points cloud by least-square fitting.

Complex shapes can only be represented by using many shape control parameters, [20], since sufficient local control of geometry is needed. In the context of interpolation, many geometric parameters link to correspondingly many internal degrees of freedom in parametric representation and consequently to high- degree curves or surfaces in 3D. Different parametric shape entities exist within the framework of geometric modeling, [3,4].

This paper develops a scheme based on chained piecewise Bezier curves and surfaces,

$$\mathbf{x}(u) = \sum_{i=0}^{n} B_{i,n}(u) \cdot \mathbf{P}_i \quad = \quad \sum_{i=0}^{n} \binom{n}{i} \cdot u^i \cdot (1-u)^{n-i} \cdot \mathbf{P}_i \tag{3}$$

where B are the basis functions (in this case Bernstein polynomials) of degree n with the parameter $u \in (0,1)$, $\boldsymbol{x}$(u) are the parametric equations of the spatial curve, and $\boldsymbol{P}$i are the $(n+1)$ control nodes. The deployment of piecewise curves provides for locality in shape representation and makes the respective chaining numerically simple to implement since $P(u)$ passes through the end nodes and has its end- slopes defined by the respective two end points.

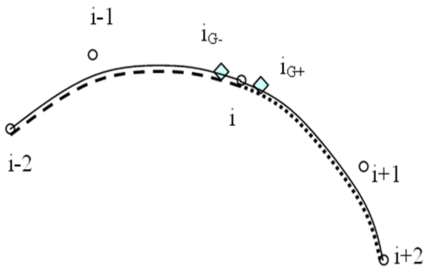


Fig. 5. Chaining of piecewise curves

The numerical procedure developed here interpolates additional points at connections of piecewise segments to impose the requested C^1 continuity, (Fig. 5),

$$P_{iG} = f(P_{i-1}, P_i, P_{i+1}) \tag{4}$$

Alternatively, B-spline curves and surfaces can be used. A B-spline curve of degree d is defined for a set of $(n+1)$ control points $\boldsymbol{P}_{\text{i}}$ as

$$\mathbf{x}(u) = \sum_{i=0}^{n} N_{i,d}(u) \cdot \mathbf{P}_i \quad , \quad u \in [0,1] \tag{5}$$

The basis functions N are defined recursively using a non-decreasing sequence of scalars- knots u_{i} such that $0 \le i \le n+d+1$ with

$$\begin{aligned} N_{i,0}(u) &= \begin{Bmatrix} 1, & u_i \le u < u_{i+1} \\ 0, & otherwise \end{Bmatrix} \quad ,0 \le i \le n+d \\ N_{i,j}(u) &= \frac{u-u_i}{u_{i+j}-u_i} N_{i,j-1}(u) + \frac{u_{i+j+1}-u}{u_{i+j+1}-u_{i+1}} N_{i+1,j-1}(u) \quad ,1 \le j \le d \,, 0 \le i \le n+d-j \end{aligned} \tag{6}$$

Analogously, B-spline surfaces are defined as

$$\mathbf{x}(u,v) = \sum_{i1=0}^{n1} \sum_{i2=0}^{n2} N_{i1,d1}(u) \cdot N_{i2,d2}(v) \cdot \mathbf{P}_{i1i2} \quad , \quad u,v \in [0,1] \tag{7}$$

where 1 and 2 denote the two directions.

In order to have the original shape of the object as the initial solution, its scanned shape (points cloud according to (2)) is the basis for fitting of geometric primitives in (3)-(7). This fitting provides the initial values of the shape optimization variables- control points in (3), (5), (7).

A simple fitting methods of a B-spline on $(m+1)$ data points $\boldsymbol{Q}_{\text{k}}$ (recorded points cloud, processed by polygonization) assumes that they are ordered with increasing sample times sequence s_{k} between 0 and 1 to correspond to parameter values. Polygonization reduces the scanned point clouds with overlapping regions to non-overlapping meshes of points of preselected density and linear faces connecting them. If the approach is 2D fitting, then planar sections can be generated from the scanned object.

2D fitting for given data points ***Q*** is implemented by determining the control points ***P*** such that the least square error

$$E(\mathbf{P}) = \frac{1}{2}\sum_{k=0}^{m}\left(\sum_{i=0}^{n} N_{i,d}(u_k)\cdot \mathbf{P}_i - \mathbf{Q}_k\right)^2 \tag{8}$$

is miminized. The minimum of the quadratic error *E* in (8), which is a function of the control points, is obtained from the necessary conditions for the respective stationary value by differentiating with respect to the control points ***P***.

If a B-spline surface is to be fitted directly on the 3D data points ***Q***, then the minimum of (9) provides the values of the control points ***P***,

$$E(\mathbf{P}) = \frac{1}{2}\sum_{k1=0}^{m1}\sum_{k2=0}^{m2}\left(\sum_{i1=0}^{n1}\sum_{i2=0}^{n2} N_{i1,d1}(u_{k1})\cdot N_{i2,d2}(v_{k2})\cdot \mathbf{P}_{i1i2} - \mathbf{Q}_{k1k2}\right)^2 \tag{9}$$

3 Shape Optimization, Development of Optimum Design Workflow

Developing an integral application encompassing all elements of Fig. 2 would create a large application, difficult to develop, maintain and use. The more reasonable option (low cost, scalability, easy maintenance, transferability of data, mild learning curve, etc) is the modular approach which uses existing applications as elements of the workflow, which is developed according to Fig. 1. The necessary development includes programs and scripts which handle the process- and data flows (including data-mining) within the workflow. In the cases presented here, an existing GA-based optimizer, [5], is upgraded and applied along with commercial FEA simulators, [8], and in-house geometric modeling tools for Bezier curves and surfaces based on (3) and (4) (alternatively B-splines (5)-(7) can be used). FEA is applied to evaluate feasibility and excellence of the candidate designs and it is invoked by the optimizer as an external service provider. The in-house developed workflow programs and scripts, [21,22], act as intermediaries and perform or monitor a number of tasks such as synchronization of asynchronous processes, data-flows and transfers, data-mining, evaluation of constraints and objectives, rejection of infeasible designs and termination of unsuccessful instances of simulators, etc.

The experimental workflow in Fig. 6 is developed by implementing an in-house GA application, ADINA FEA simulator and in-house developed middleware (developed in the *C#* language with *.net* 3.5 and the Windows Communication Foundation *WCF*), while native ASCII files are used for data storage.

The client implements the optimizer based on GA, [23,24], and the basic objective function framework which engages the services of many networked server computers, [25], which run listener programs and expose FEA simulators services. When called by the client, the respective server listener program invokes the execution of the locally installed instance of the FEA simulator on the respectively assigned input files and produces output files correspondingly. The client data- mines the corresponding output files for the results needed, such as stresses at the Gauss quadrature points of all elements.

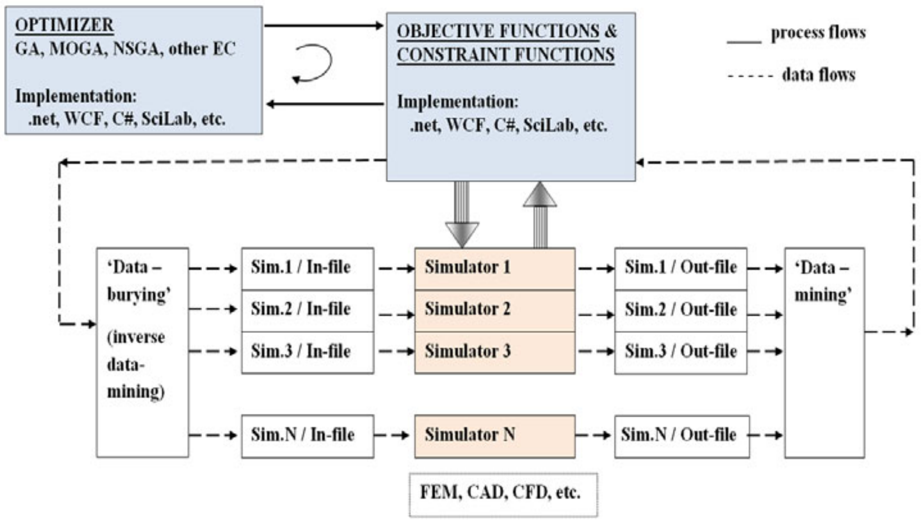


Fig. 6. Set-up of the sequential- parallel experimental workflow for optimum design

The networked servers exposing the FEA (and other) services are invoked parallely in the asynchronous mode by the client. The client can also request partial services of the applications running at the servers, as the requests are communicated by message-passing or sendkeys sequences. Unsucessful instances of services not producing the output files or those producing incorrect outputs are identified, dismissed and terminated by the client.

The *Service* class running under *.net* and *WCF* uses the *ServiceHost* class to configure and expose a service for use by client applications, *Binding* and service *Endpoints*, *ServiceContract* class to specify messages in the conversation, *OperationContract* class to define operations within a service contract in *WCF*, *Process* class to encapsule locally running applications at servers, *SendKeys* class to pass messages to remote applications, *FindWindow, SetForegroundWindow* and *SetActiveWindow* APIs to access basic Windows functionality, *IntPtr* handles to remotely access processes which host applications and windows, and *Delegate* references which point to methods and provide for asynchronus execution of services.

4 Test Example

The test case presented here is a simple 2D problem (Fig. 7) for which the respective optimum shape solution is known. This simple standard example is applied to demonstrate the application of the procedure and the workflow. In particular, a 2D plate with a hole in plane stress conditions is loaded bi-axially (2:1 loading ratio) with minimum mass as the excellence criterion and subject to the constraint of the maximum permissible (local) values of the Von Mises stress, Fig. 7.

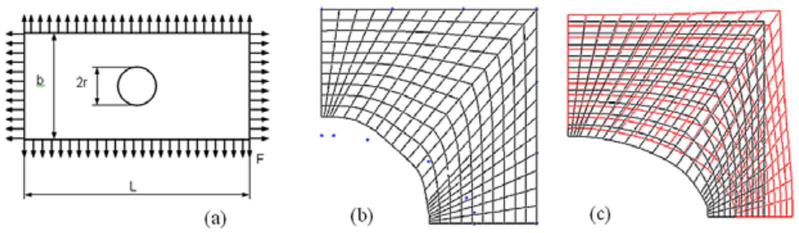


Fig. 7. Plane stress plate (a) with a hole, (b) model and initial shape, (c) optimized shape with displacements

The mass and stresses values are data- mined from the output files of the FEA simulator, (Fig. 6). The mass of the candidate design in each iteration of the shape optimizer is evaluated from the corresponding candidate design geometry such that the areas of all the finite elements in the FE mesh are aggregated. The stresses are data- mined for all Gauss integration support points for all finite elements of the mesh, which provides for evaluation of both average and extreme values that can consequently be used in the constraint equations and penalized.

The phenotype in Fig. 2 implies all the parameters that define the geometry of the candidate design, which also includes boundaries which are fixed (or partially fixed by boundary conditions) and therefore not subject to shape optimization. On the other hand, the genotype only codes the free design variables of the geometry. In the test case in Fig. 7, the radii of the control points at prescribed angular increments and alternatively y-coordinates for prescribed x-increments were used as free variables and coded in the genotype. Decoding the genotype and combining it with fixed values provides the phenotype that fully defines the piecewise Bezier curve (3) which models the shape of the free boundary to be shape-optimized, and which is updated into the corresponding FEA input file for each candidate design, Fig. 6.

The external penalty formulation was used with 5-9 shape variables, (3), GA populations of up to 100 and up to three elite members, rank- based scaling, uniform selection, scattered cross-over with probability 0.8, Gaussian mutation, and standard values of other parameters. Normalized minimum mass was combined with penalized normalized Von Mises stresses. Other GA operators, for example proportional scaling, roulette selection, heuristic cross-over and different probabilities for cross-over and mutation were applied as well, but this did not result in major differences in the convergence of the process.

Table 1 presents the impact of parallelization (Fig. 6) of the FEA simulators, whereby the central GA optimization procedure hosted at the client computer is served by up to 18 FEA locally installed simulators at server computers and invoked by listener programs running on the server computers. The combinations of the numbers of engaged server computers and sizes of populations were selected such that the inpact of paralelization could be benchmarked. These combinations are illustrative in terms of the number of individuals of some GA population per server computer which corresponds to the number of FEA simulator runs (and data mining operations) per server for each GA generation.

Table 1. Benchmarking of the impact of parallelization with the experimental workflow in Fig. 6, test case in Fig. 7, normalized relative execution times

Population size / Number of servers	Optimization execution time
6 / 1	5.56
12 / 1	11.01
18 / 1	16.16
6 / 6	1
12 / 12	1
18 / 18	1

The ratios in Table 1 can also be extended to more parallel server computers engaged in the workflow. The table clearly indicates that the optimization execution times are close to proportional to the ratio of population size per number of servers, which corresponds to the number of FEA runs to be sequentially executed by each server computer for each GA generation. In the cases where the population size is not a multiple of the number of servers, there would exist an unbalanced workload for the individual servers, which would cause wait (idle) times for less busy servers before the next generation of GA individuals is submitted to them. Table.1 also indicates that the central optimization procedure (GA algorithm), obviously time-wise dominated by the FEA simulations, does not contribute significantly to the overall execution time.

The main objective of this paper is to demonstrate how an efficient generic workflow for evolutionary shape optimization can be built inexpensively using existing program components such as FEA simulators. The workflow developed in this paper includes custom-made programs for parameterization, client-server communication to enable remote execution, asynchronous execution for parallelization, and data- mining scripts for the transfer and localization of data. The shape optimization procedure is improved by developing a procedure to use an existing design as the initial solution by 3D scanning and parameterizing its geometry. The workflow essentially implements the elements of Fig. 1 and Fig. 2 in an integrated manner.

5 Conclusion

An experimental workflow for shape optimization problems is developed along with a PC cluster implementation for sequential- parallel execution. The workflow employs commercial applications for optimization and finite element analysis, while the shape parameterization and representation is developed based on chained Bezier curves, while alternatively B-splines can also be used. The necessary workflow control scripts and programs are developed in C# within *MS .net* and *WCF* platforms. The results demonstrate this to be a viable, flexible and inexpensive framework for shape optimization. The initial solution for shape optimization can be the optically scanned shape of an existing object after triangulation, polygonization and parameterization which make this approach an enhanced reverse engineering system based on GA.

Acknowledgments

This work was supported in part by the Croatian Ministry of Science, Education and Sports, project number 023-0231744-3113, Intelligent and Evolutionary Algorithms in Optimization.

References

[1] Arora, J.: Introduction to Optimum Design. McGraw-Hill, New York (1989)
[2] Rao, S.S.: Engineering Optimization. Wiley Interscience, Hoboken (1996)
[3] Farin, G.: Curves and Surfaces for Computer Aided Geometric Design. Academic Press, London (1993)
[4] Bohm, W., Farin, G., Kahmann, J.: A survey of curve and surface methods in CAGD. Computer Aided Geometric Design 1, 1–60 (1984)
[5] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading (1989)
[6] Deb, K., Goel, T.: Multi-Objective Evolutionary Algorithms for Engineering Shape Design. KanGAL report 200003, Indian Institute of Technology (2000)
[7] Cook, R.: Concepts and Applications of Finite Element Analysis. John Wiley & Sons, Chichester (1989)
[8] Adina: ADINATM 8.3 User interface and command reference manual. Adina R&D Inc. (2005)
[9] Saitou, K., Izui, K., Nishiwaki, S., Papalambros, P.A.: Survey of Structural Optimization in Mechanical Product Development. Transactions of the ASME 5, 214–226 (2005)
[10] Cyganek, B., Siebert, J.P.: An Introduction To 3D Computer Vision Techniques And Algorithms. John Wiley & Sons, Chichester (2009)
[11] Peng, T., Gupta, S.K.: Model and algorithms for point cloud construction using digital projection patterns. ASME Journal of Computing and Information Science in Engineering 7(4), 372–381 (2007)
[12] Sarkar, B., Menq, C.H.: Smooth-surface approximation and reverse engineering. Computer-Aided Design 23(9), 623–628 (1991)
[13] GOM, http://www.gom.com/metrology-systems/system-overview/atos.html
[14] Eberly, D.: Least-Squares Fitting of Data with B-Spline Surfaces, Geometric Tools, LLC, http://www.geometrictools.com/
[15] Eck, M., Hoppe, H.: Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. In: ACM SIGGRAPH 1996 Conference Proceedings, pp. 325–334 (1996), http://research.microsoft.com/en-us/um/people/hoppe/proj/bspline/
[16] Kruth, J.P., Kerstens, A.: Reverse engineering modelling of free-form surfaces from point clouds subject to boundary conditions. Journal of Materials Processing Technology 76, 120–127 (1998)
[17] Ma, W., Kruth, J.P.: NURBS curve and surface fitting for reverse engineering. International Advanced Manufacturing Technology 14(12), 918–927 (1998)
[18] Ristic, M., Brujic, D.: Efficient registration of NURBS geometry. Image Vision Comput. 15, 925–935 (1997)
[19] Piegl, L.A., Tiller, W.: Parametrization for surface fitting in reverse engineering. Computer-Aided Design 33, 593–603 (2001)

[20] Samareh, J.A.: A Survey Of Shape Parameterization Techniques. In: CEAS/AIAA/ICASE/ NASA Langley International Forum on Aeroelasticity and Structural Dynamics, NASA/ CP- 1999-209136, pp. 333–343 (1999)

[21] Resnick, S., Crane, R., Bowen, C.: Essential Windows Communication Foundation for.NET Framework 3.5. Addison Wesley, Reading (2008)

[22] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes. Cambridge University Press, Cambridge (1992)

[23] De Jong, K.A., Spears, W.M.: An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 38–47. Springer, Heidelberg (1991)

[24] Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control In Evolutionary Algorithms. IEEE Transactions On Evolutionary Computation 3(2), 124–141 (1999)

[25] Alonso, J.M., Alfonso, C., Garcıa, G., Hernandez, V.: GRID technology for structural analysis. Advances in Engineering Software 38, 738–749 (2007)

Fault-Tolerance Simulation of Brushless Motor Control Circuits

Huicong Wu[1,2,3], Jie Chu[2], Liang Yuan[2], Qiang Zhao[2], and Shanghe Liu[2]

[1] School of Information Science and Engineering
Hebei University of Science and Technology,
Shijiazhuang 050018, China
[2] Institute of Electrostatic and Electromagnetic Protection
Ordnance Engineering College,
Shijiazhuang 050003, China
[3] CERCIA, School of computer science, The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
whc@hebust.edu.cn

Abstract. Brushless Motors are frequently employed in control systems. The reliability of the brushless motor control circuits is highly critical especially in harsh environments. This paper presents an Evolvable Hardware (EHW) platform for automated design and adaptation of a brushless motors control circuit. The platform uses the principles of EHW to automate the configuration of FPGA dedicated to the implementation of the motor control circuit. The ability of the platform to adapt to a certain number of faults was investigated through introducing single logic unit faults and multi-logic unit faults. Results show that the functionality of the motor control circuit can be recovered through evolution. They also show that the location of faulty logic units can affect the ability of the evolutionary algorithm to evolve correct circuits, and the evolutionary recovery ability of the circuit decreases as the number of fault logic units is increasing.

Keywords: Evolutionary Algorithms, Evolvable Hardware, Fault Tolerance, Motor Control Circuits.

1 Introduction

Brushless motors are frequently employed in the speed regulation of many driving systems. The performance and sustained reliability of the motor control circuit are of great importance. Usually the control circuits designed in SCM or DSP are easily damaged in extreme environmental conditions, such as electromagnetic interference and high-energy radiation.

Recently, fault tolerant systems are widely used in space applications where hardware deteriorates due to damages caused by aging, temperature drifts and high-energy radiation. In this case, human intervention is difficult or impossible; the systems must therefore maintain functionality themselves. Conventional fault tolerant systems employ techniques such as redundancy, checking-pointing and

© Springer-Verlag Berlin Heidelberg 2011

concurrent error detection. These techniques all rely on the presence of additional redundancy and add considerable cost and design complexity. In most cases, it can't satisfy the application requirements [1].

As a promising research field, Evolvable Hardware (EHW) [2–4] may provide alternative approaches and new mechanisms for the design of fault tolerant systems. EHW is based on the idea of combining reconfigurable hardware devices with an evolutionary algorithm (EA) to perform reconfiguration autonomously [3], which refers to the characteristics of self-organization, self-adaptation and self-recovery. With the use of evolutionary computation, evolvable hardware has the capability of autonomously changing its hardware architectures and functions. It can maintain existing function in the context of degradations or faults in conditions where hardware is subject to faults, temperature drifts, high-energy radiation, or aging.

As to logic or digital circuits, gate-level evolution usually takes logic gates as the basic units or building-blocks. Many researchers in this field prefer extrinsic evolution at gate-level [5] because it is generally applicable to various circuits and its outcomes are comparatively formal and consequently analyzable. Many encouraging results for gate-level evolution of logic circuits have been demonstrated [5–7]. Nanjing University of Aeronautics and Astronautics performed the online fault tolerant evolution of digital circuits and analogy circuits on FPGA and FPTA respectively [8–10]. The Jet Propulsion Laboratory (JPL) performs research in fault tolerant and space survivable electronics for the National Aeronautics and Space Administration (NASA). JPL has had experiments to illustrate evolutionary hardware recovery from degradation due to extreme temperatures and radiation hardware environments [11, 12]. Their experiment results demonstrate that the original functions of some evolved circuits, such as low-pass filters and the 4-bit DAC, could be recovered by reusing the evolutionary algorithm that altered the circuit topologies [11, 12].

This paper presents an evolvable hardware platform for the automated design and adaptation of brushless motor control circuits. The platform employs an EA to autonomously configure the FPGA dedicated to the implementation of the motor control circuit. The ability of the platform to adapt to a certain number of faults was investigated through introducing single logic unit faults and multi-logic unit faults.

2 Fault Tolerant Platform

The brushless motor achieves the phases changing operation with electronic circuit. The system structure is illustrated in Fig. 1. It includes three parts, the motor control circuit, the drive module and the brushless motor itself [13, 14]. The brushless motor checks the position of the rotors by using 3 location sensors. It produces three position feedback signals S_0, S_1 and S_2. When the Rotor rotates 360 degrees along the same direction, the position signal S_0, S_1 and S_2 have a total of six states combination. The motor control circuit triggers each switch $(M_0, M_1, M_2, M_3, M_4, M_5)$ in the drive module in accordance with a certain order.

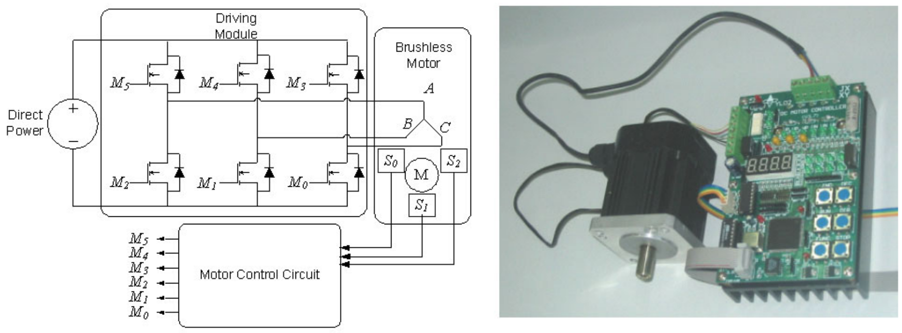


Fig. 1. The motor control system [13]

The motor control circuit fault tolerant evolutionary environment is shown in Fig. 2. The platform comprises of FPGA, EA evolutionary module, VHDL coding conversion module and FPGA development tool software environment.

Alter EP1K50 FPGA, which is capable of partial dynamic reconfiguration, was adopted as the experiment hardware. It provides a Joint Test Action Group (JTAG) system interface connected to the computer parallel port, through which the circuit configuration bits can download to FPGA to validate its functionality.

The evolutionary module is the core part of the system. Circuit structure is represented by a chromosome. The simulated evolution is used to evolve a good set of architecture bits that determine the functions and interconnections of the logic units in FPGA.

The VHDL coding conversion module together with Quartus II can realize the conversion from the chromosome representation to a circuit structure.

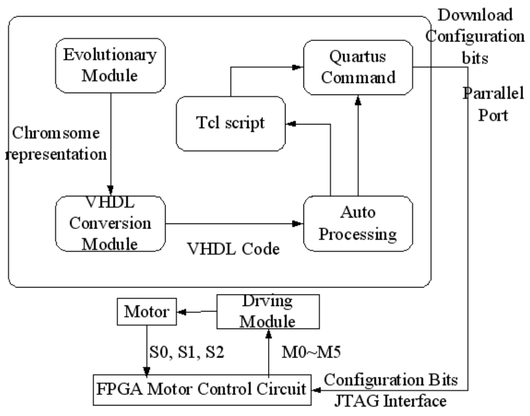


Fig. 2. EHW platform for the motor control circuit fault recovery implementation

3 Evolutionary Circuit Design

Evolutionary algorithms are used for the brushless motor control circuit design. Circuit representation, fitness evaluation, and parameters choice are crucial ingredients of effective evolutionary circuit design [4].

3.1 Chromosome Representation

A correct circuit representation is the base for effective design. Fig. 3 shows the computational model for gate-level evolution of the brushless motor control circuit. The evolution area is an array of 8*5. Because the first column works as inputs and the last as outputs, the two columns won't participate in the evolution. The actual evolutionary area is the form of a rectangular array that consists of logic units in 8 rows by 3 columns. Each logic unit has 2 inputs, one output and can perform 4 functions: AND, OR, NAND, NOR.

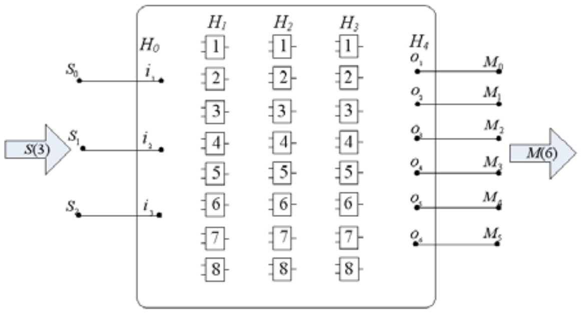


Fig. 3. The computational model of motor control system

The configuration array which represents interconnections and functions of the logic units is shown as following:

$$C_{0,1} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,l} \\ b_{1,1} & b_{1,2} & \dots & b_{1,l} \\ w_{1,1} & w_{1,2} & \dots & w_{1,l} \\ w_{2,1} & w_{2,2} & \dots & w_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,l} \end{bmatrix} . \quad (1)$$

$$C_{k-1,k} = \begin{bmatrix} a_{k,1} & a_{k,2} & \dots & a_{k,l} \\ b_{k,1} & b_{k,2} & \dots & b_{k,l} \\ w_{1,1} & w_{1,2} & \dots & w_{1,l} \\ w_{2,1} & w_{2,2} & \dots & w_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ w_{l,1} & w_{l,2} & \dots & w_{l,l} \end{bmatrix} \quad (2 \le k \le K-1) . \quad (2)$$

$$C_{K-1,K} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{l,1} & w_{l,2} & \dots & w_{l,n} \end{bmatrix} . \tag{3}$$

The configuration array includes two parts, the functional array expressed as $a_{i,j}$ in the first two rows and the connectivity array expressed as $w_{i,j}$ in the rest rows. Each logic unit comprised 4 functions can be encoded in binary column vector format $(a_{k,i}, b_{k,i})^T$, As the output interfaces, there is no corresponding function array in column H_4.

In formulation (1), $C_{0,1}$ represents the configuration array between column H_0 and H_1. Here the value of m is 3 according to 3 inputs in column H_0; the value of l is 8 according to 8 logic units in column H_1. $w_{i,j}$ represents the connection relationship of logic unit in column H_1 with each logic unit in previous column H_0. the value of $w_{i,j}$ is '1' or '0'. In formulation (2), $C_{k-1,k}$ represents the configuration array between columns H_{k-1} and H_k. Here the value of K is 4 according to the maximun column number H_4. The value of l is 8 according to 8 logic units in each column H_1, H_2 and H_3. In formulation (3), $C_{K-1,K}$ represents the connectional array between columns H_{K-1} and H_K. the value of n is 6 according to the 6 bit outputs.

3.2 Fitness Evaluation

For problems of gate-level evolution, design objectives mainly include expected functions, efficiency of resource usage (in terms of gate count) and operating speed of circuits (estimated with Maximal Propagation-Delay (MPD)). Although a functionally correct circuit with fewer logic gates and fewer number of gates contained in the longest signal chain of the circuit is usually preferable, the main purpose in this paper is to investigate the capacity of fault recovery using EHW in case of faults. Therefore, the design objective only concerns with the expected functions or behaviors. Thus, the functional fitness value of the evolved circuit is calculated as

$$F = \sum_{i=1}^{n} \sum_{j=1}^{m} C_{i,j} \qquad C_{i,j} = \begin{cases} 1 & outdata = epdata \\ 0 & outdata \neq epdata \end{cases} \tag{4}$$

where *outdata* is output value of currently evaluated circuit; *epdata* is output value of expected circuit.

3.3 Adaptation Strategy for EA Parameters

Some EA parameters, especially P_c and P_m, have large effects on EA's performances; and their optimal values are usually impossible to be predefined to suit various problems[6, 15]. In our approach, P_c and P_m are varied with the individuals' distribution and EA's genetic processes so as to maintain diversity in the population and sustain the convergence capacity of the EA. Diversity in the population is estimated by using

$$\delta_t = e^{\frac{f_{max} - f_{min}}{f_{avg}}} \qquad (0 < \delta_t < 1) \ . \tag{5}$$

P_c and P_m are designed to adapt themselves in the following ways

$$P_c = \begin{cases} P_{c0} & 0 < \delta_t \leq k1 \\ P_{c1} + \frac{(P_{c0} - P_{c1})(1-\delta_t)}{1-k1} & k1 < \delta_t < 1 \end{cases} \ . \tag{6}$$

$$P_m = \begin{cases} P_{m0} & 0 < \delta_t \leq k2 \\ P_{m0} + \frac{(P_{m1} - P_{m0})(\delta_t - k2)}{1-k2} & k2 < \delta_t < 1 \end{cases} \ . \tag{7}$$

where, P_{c0} and P_{m0} are initial values of P_c and P_m respectively, it is usually feasible to let $P_c = 0.8$ and $P_m = 0.1$ due to the above adaptation strategy; k_1 and k_2 are user-defined constants, we chose to let $k_1 = k_2 = 0.3$. According to the above equations, P_c and P_m will respond to changes of individuals' diversity reflected by δ_t.

4 Fault-Tolerance Experiments

The objective of the experiments was to recover the functionality of the motor control circuit implemented on an FPGA. In this experiment, faults were introduced by setting all connections with the corresponding fault logic unit to '0'. Different numbers of faults were introduced for experiments. Firstly, we evolved a motor control circuit in case all 24 logic units available; Secondly, single faults in different position of the circuit and multi-faults in column H_2 were introduced respectively; and then evolutionary process was carried out to recover the circuit topology with the same functionalities. In each evolutionary process, the program stopped when the best individual's fitness was 36 or the generations were 3000.

4.1 None Fault Experiment

In case of 24 logic units available, 50 runs of the program were executed in our evolutionary platform. The experiment results are shown in Table 1. One of the evolutionary processes is shown in Fig. 4; The corresponding evolved circuit is depicted in Fig. 5.

Table 1. Experimental results of 50 runs

Criterion	Generation	Fitness	Evolution Time (s)	Number of Logic Unit
Average Values	493.4	30.67	96.72	17.3
Standard Deviation	95.72	2.29	22.67	0.52

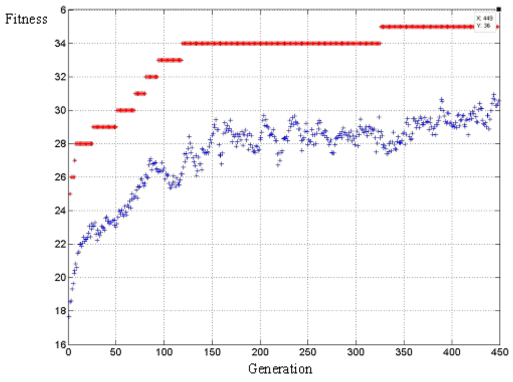


Fig. 4. Diagram of the population fitness. The top line is the best individual's fitness at each generation and the bottom is the average fitness.

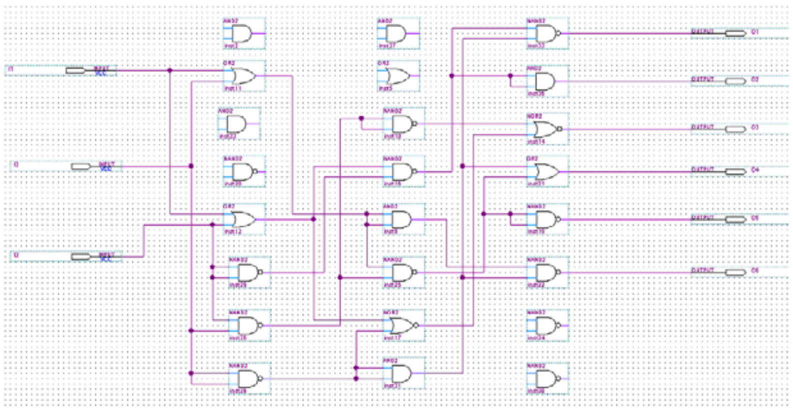


Fig. 5. The evolved motor control circuit structure

4.2 Single Logic Unit Fault

The aim is to test that the platform has good fault recovery ability for single logic unit fault. When fault is introduced to logic unit in H_2 column, the recovery rate is 100%; that is to say, the circuit evolved can recover from single logic unit fault completely. But when fault is introduced to H_1 and H_3 column, the correct circuit can't be evolved correctly all the time after 3000 generations evolution. Furthermore the average fitness decreases and the average evolutionary generations increase. That is because the fault unit is near the inputs and outputs position; the location of fault logic unit has crucial impact on fault recovery ability. It will greatly affect the ability of the EA to evolve high quality circuit in limited generations. Faults close to the inputs or outputs will have a more detrimental effect than those distributed in the centre column. Table 2 summarizes the experimental results with a single fault introduced.

Table 2. Experimental results with single fault introduced

Position of Fault	Average Generation	Average Fitness	Evolution Time (s)	Recovery Rate	Number of Logic Unit
H_2	506.3	30.53	96.42	100%	17.4
H_1	823.4	27.46	183.67	90%	16.5
H_3	1137.8	28.17	251.30	70%	18.7

4.3 Multi-logic Unit Fault

Here increasing number of logic unit faults in H_2 column are introduced to illustrate fault-tolerance ability of the motor control circuit respectively. The experiment results are shown in Table 3.

Table 3. Experimental results with increasing numbers of fault introduced

Number of Fault	Average Generation	Average Fitness	Evolution Time (s)	Recovery Rate	Number of Logic Unit
1	506.3	30.53	98.42	100%	17.6
2	737.8	27.46	172.67	100%	16.5
3	1258.4	28.17	281.30	70%	16.4
4	2475.3	27.12	475.42	30%	15.2
5	3000.0	20.43	709.57	0%	-

Table 3 indicates that the fault tolerant ability of FPGA decreases as the number of fault logic units is increasing. Especially, When four logic unit faults occur, the recovery rate is 30%; the average fitness diminishes and the average number of evolutionary generations increase rapidly. The average number of logic units used to implement the circuit reduces as the number of faults increases.

From the experimental results above, we know that the number of fault logic units is closely related to the fault tolerant ability; that is to say, with the number of fault logic units increasing, evolutionary recovery of the same circuit needs more evolutionary generations, and the average fitness and recovery rate decrease evidently. The reason is that the increasing number of fault logic units makes the signal paths which are used to accurately transfer signals become less. consequently, to evolve the objective circuit topologies become more difficult. We also find that if 5 logic units cause faults, the correct functional circuit can't be evolved; that is to say, the most permissive faults are 4 logic units.

An example configuration of the motor control circuit evolved with 4 logic unit faults is illustrated in Fig. 6. It is emphasized that the objective of this work was not explicitly to design more efficient circuits but to show that it is possible to evolve an alternative circuit in case of fault occur in the original circuit, so that the functionality can be recovered.

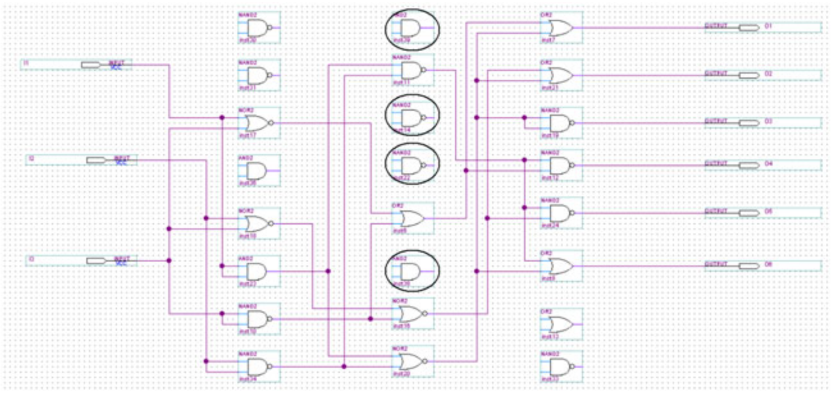


Fig. 6. The evolved motor control circuit with four logic unit faults introduced

5 Conclusion

A fault tolerant hardware platform for the automated design of brushless motor control circuit has been presented. The platform uses the principle of EHW to automate the configuration of FPGA dedicated to the implementation of the motor control circuit. Our experiments show that it is possible to recover the function of motor control circuit through evolution when faults are introduced. Furthermore, the ability of the platform to adapt to increasing numbers of faults was investigated. Results show that the functional circuit can be derived from single logic unit faults and multi-logic unit faults; the most permissive faults are four logic units. Of course, the location of faulty logic units will influence the ability of EA to evolve high quality circuits, faults directly on logic units which are connected to the inputs and outputs will have a more detrimental effect than those distributed in the centre of the topological structure. This is similar to an earlier observation [16]. It also shows that the evolutionary recovery ability of the motor control circuit decreases as the number of fault logic units increasing.

The real attractiveness and power of EHW comes from its potential as an adaptive hardware while operating in a real physical environment [4, 17]. Further work will focus on on-line evolution in electromagnetic interference environments, which poses a great challenge, although online learning approaches [18] can be employed in our EHW system.

References

1. Arslan, B.I., Thomsom, T.R.: Evolutionary Design and Adaptation of High Performance Digital Filters with an Embedded Reconfigurable Fault Tolerant Hardware Platform. In: Software Computing, vol. 8, pp. 307–317. Springer, Berlin (2004)
2. Higuchi, T., Niwa, T., Tanaka, T., de Garis, H., Furuya, T.: Evolvable Hardware with Genetic Learning: A first step toward building a Darwin machine. In: Proc. of Second International Conference On the Simulation Adaptive Behavior (SAB 1992), Cambridge, MA, pp. 417–424 (1992)

3. Higuchi, T., Liu, Y., Yao, X. (eds.): Evolvable Hardware. Springer Science+Media LLC, New York (2006)
4. Yao, X., Higuichi, T.: Promises and Challenges of Evolvable Hardware. IEEE Trans. On Systems Man and Cybernetics-Part C: Applications and Reviews 29, 87–97 (1999)
5. Murakawa, M., Yoshizawa, S., Kajitani, I., Yao, X., Kajihara, N., Iwata, M., Higuchi, T.: The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing. IEEE Transactions on Computers 48(6), 628–639 (1999)
6. Zhao, S.G., Jiao, L.C.: Multi-objective Evolutionary Design and Knowledge Discovery of Logic Circuits Based on an Adaptive Genetic Algorithm. In: Genetic Programming and Evolvable Machines, vol. 8, pp. 195–210. Springer, Berlin (2006)
7. He, J., Yao, X., Chen, Y.: A Novel and Practicable On-chip Adaptive Lossless Image Compression Scheme Using Intrinsic Evolvable Hardware. Connection Science 19(4), 281–295 (2007)
8. Gao, G.J., Wang, Y.R., Cui, J., Yao, R.: Research on Multi-objective On-line Evolution Technology of Digital Circuit Based on FPGA Model. In: Proc. of 7th International Conference of Evolvable System: From Biology to Hardware, Wuhan, China, pp. 67–76 (2007)
9. Ji, Q.J., Wang, Y.R., Xie, M., Cui, J.: Research on Fault-Tolerance of Analog Circuit Based on Evolvable Hardware. In: Proc. of 7th International Conference of Evolvable System: From Biology to Hardware, Wuhan, China, pp. 100–108 (2007)
10. Yao, R., Wang, Y.R., Yu, S.L., Gao, G.J.: Research on the Online Evolution Approach for the Digital Evolvable Hardware. In: Proc. of 7th International Conference of Evolvable System: From Biology to Hardware, Wuhan, China, pp. 57–66 (2007)
11. Stoica, A., Keymeulen, D., Arslan, T., Duong, V., Zebulum, R.S., Ferguson, I., Guo, X.: Circuit Self-Recovery Experiments in Extreme Environments. In: Proceedings of the 2004 NASA/DoD Conference on Evolution Hardware, Seattle, WA, USA, pp. 142–145 (2004)
12. Stoica, A., Keymeulen, D., Zebulum, R.S., Thakoor, A., Daud, T., Klimeck, G., Jin, Y., Tawel, R., Duong, V.: Evolution of Analog Circuits on Field Programmable Transistor Arrays. In: Proc. of the Second NASA/DOD Workshop on Evolvable Hardware, pp. 99–108. IEEE Computer Society Press, Los Alamitos (2000)
13. Yuan, L., Ding, G.L., Liu, W.B., Huang, F.Y., Zhao, Q.: TMR Hardware and Its Implementation on Controlling System. Computer Engineering 32(21), 249–251 (2006)
14. Chu, J.: Study of the Fault Tolerant Bionic circuit Model. PhD. Dissertation. Ordnance Engineering College, Shijiazhuang, China (2009) (in Chinese)
15. Zhao, S.G.: Study of the Evolutionary Design Methods of Electronic Circuits. PhD. Dissertation. Xidian University, Xi_an, China (2003) (in Chinese)
16. Schnier, T., Yao, X.: Using Negative Correlation to Evolve Fault-Tolerant Circuits. In: Tyrrell, A.M., Haddow, P.C., Torresen, J. (eds.) ICES 2003. LNCS, vol. 2606, pp. 35–46. Springer, Heidelberg (2003)
17. Yao, X.: Following the Path of Evolvable Hardware. Communications of the ACM 42(4), 47–49 (1999)
18. Minku, L.L., White, A., Yao, X.: The Impact of Diversity on On-line Ensemble Learning in the Presence of Concept Drift. IEEE Transactions on Knowledge and Data Engineering 22(5), 730–742 (2010)

Parallel Evolutionary Optimization of Digital Sound Synthesis Parameters

Batuhan Bozkurt[1] and Kamer Ali Yüksel[2]

[1] Centre for Advanced Studies in Music,
Istanbul Technical University, Macka, 34357 Istanbul, Turkey
batuhan@batuhanbozkurt.com
[2] Computer Vision and Pattern Analysis Laboratory,
Sabanci University, Orhanli - Tuzla, 34956 Istanbul, Turkey
kamer@sabanciuniv.edu

Abstract. In this research, we propose a novel parallelizable architecture for the optimization of various sound synthesis parameters. The architecture employs genetic algorithms to match the parameters of different sound synthesizer topologies to target sounds. The fitness function is evaluated in parallel to decrease its convergence time. Based on the proposed architecture, we have implemented a framework using the SuperCollider audio synthesis and programming environment and conducted several experiments. The results of the experiments have shown that the framework can be utilized for accurate estimation of the sound synthesis parameters at promising speeds.

Keywords: computer music, parameter estimation, evolutionary computation, parallel computing.

1 Introduction

Any attempt for sound analysis is also a form of endeavor for some sort of parameter estimation [16, pg 596]. The analysis task might be undertaken for obtaining the properties of some source sound that is to be re-synthesized with different sound synthesis methods, or for observing those very qualities with the purpose of fitting them into a theoretical model. For instance, Roads [16, pg 596] points out that the Fourier Analysis can be considered as a parameter estimation method, because the results returned by such an analysis (namely magnitude and phase dissections for the analyzed signal) can be considered as parameters for a sine wave re-synthesis method that will approximate the source content veridically. However, we approach the problem of parameter estimation for effectively reducing the amount of data that is required to approximate a given sound with different synthesis methods in order to be able to control and alter various perceptual qualities of the resulting sounds from a higher level of abstraction, in an intuitive and interactive manner.

In this paper, we introduce the use of a parallelizable evolutionary architecture to optimize the parameters of sound synthesizers. The architecture is implemented as a modular evolutionary framework conceived inside the SuperCollider

© Springer-Verlag Berlin Heidelberg 2011

(SC) programming language that specializes in audio synthesis and algorithmic composition [12,13]. The framework uses genetic algorithms (GA) to automatically optimize the set of parameters required to approximate any given target sound, using an arbitrary sound synthesizer topology created by the user. In order to test the efficiency of the framework, we have experimented with a percussion and a multiple modulator frequency modulation synthesizer for various target sounds. Finally, we have described ideas for opportunities on creative usages of evolutionary methodologies, which perform at interactive speeds in a highly connective real-time sound synthesis and algorithmic composition environment. The primary contribution of this work is the promising convergence time, which is obtained through the parallel architecture implemented using the SC environment and the simplified fitness function that preserves the perceptual quality.

2 Related Works

Evolutionary methodologies have been previously investigated by researchers to solve the problem of tone matching and parameter estimation for several different synthesizer topologies. Manzolli et al. [11] introduced the evolution of waveforms to the corresponding psychoacoustic attributes of target sounds. Horner et al. utilized evolutionary methods for parameter matching using Wavetable Synthesis [6], Frequency Modulation Synthesis (FM) [9,7,8], and Group Synthesis [2]. Garcia [5] used a genetic programming approach for automating the design of sound synthesizer algorithms represented in the form of acyclic tree graphs. Johnson [10] benefited from interactive genetic algorithms (IGA) where an user conducted interactive approach was investigated to search the parameter space and direct the parameters of the Csound FOF synthesizer.

The vast majority of the studies in this area aim to justify the suitability and efficiency of the proposed methods for the aforementioned task. For that reason, the software tools used in those works are quite specialized in demonstrating the traits of the particular proposition in which the research focuses upon (with the exception of [1] and [4] using Csound and Pure Data). Consequently, they lack connectivity with other pieces of digital music performance software; thus, obtaining and using them for compositional, live performance and other creative purposes in a practical manner is difficult.

3 Architecture

The proposed architecture consists of a host environment for sound synthesis, a modular evolutionary framework (EVWorkBench) and a parameter estimation system (EVMatch).

The host environment is implemented using [13] audio synthesis and programming language of SC (referred as *sclang*). The sound synthesis requests are handled at server-side (*scsynth*) within the SC environment through compiled unit generator graph functions (Fig. 1). This architecture allows a single sclang client to control multiple instances of local and networked scsynth servers via the Open Sound Control (OSC) protocol [17].

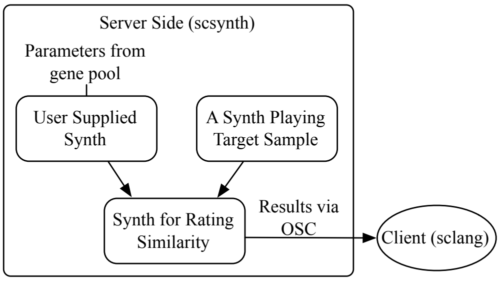

Server Side (scsynth)
Parameters from gene pool
User Supplied Synth
A Synth Playing Target Sample
Synth for Rating Similarity
Results via OSC
Client (sclang)

Fig. 1. Server-side Fitness Evaluation through OSC protocol

The initialization of the parameters and application of genetic operators are handled by the wrapped EVWorkBench class. In this way, the stages of the optimization workflow (such as initialization, selection, crossover and mutation) are separated to create a modularized framework where various evolutionary approaches can easily be experimented. Furthermore, the layered learning (having increasingly demanding fitness functions) [14] and fine tuning of the bounds of mutation operators throughout evolution are supported through interpreted and interactive language of the host environment

The EVMatch class is responsible for compilation of the synthesis definition (referred as SynthDef) provided by the user, transmission of the compiled SynthDef to the supplied servers (local and networked), the evaluation and optimization of the parameters in the gene pool, and distribution of the computational load of the evaluation across the servers registered with the instance.

In this work, the optimization is performed inside the parameter spaces of static synthesizer topologies supplied by the user for a target sound. However, it is also possible to encode synthesizer topologies as chromosomes and evolve sound synthesis algorithms as described by [5]. The speed of optimization can be increased by affecting the convergence time of the search using the direct control of the selection pressure (e.g. the tournament size) that can be tuned for different search domains and desired time constraints.

3.1 GA Optimization

To begin optimization, the user supplies a target sound file with desired attributes, a synthesizer definition, parameter names and default values of the parameters, which forms the initial population. After this initialization, the GA loop (Fig. 2) happens in generations for evolving towards better solutions of parameter sets. In each iteration, GA synthesize sounds for each set of parameter and compare the attributes of the output sound to the target sound by

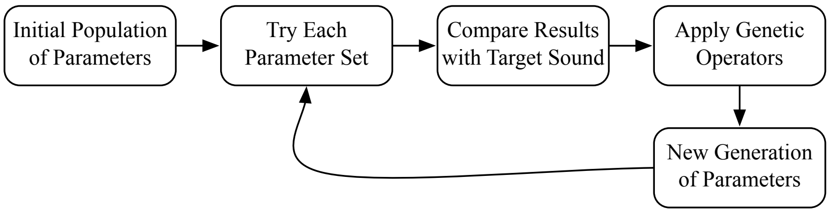

Initial Population of Parameters
Try Each Parameter Set
Compare Results with Target Sound
Apply Genetic Operators
New Generation of Parameters

Fig. 2. The GA Loop

calculating the fitness of them. Then, multiple individuals are stochastically selected from the current population based on their fitness to breed a new generation. After selection, crossover and mutation operators are applied to the gene pool. This loop continues until satisfactory fitness level, which is set by the user, has been reached for the population. In addition to a target fitness value, the user can decide the fitness of the fittest member of the last generation by listening or can limit the maximum number of generations to be iterated.

Reproduction. The tournament selection [15] is preferred concerning the relationship between the convergence rate of the GA and selection pressure, and suitability of tournament selection for noisy GA chains. Tournament selection involves running several tournaments among a few individuals chosen at random from the population. After each tournament, the one with the best fitness (the winner) is selected for crossover. Afterwards, a multi-point crossover operator where the number of split points determined proportionally to the crossover probability is applied.

4 Fitness Evaluation

The fitness evaluation provides fitness scores for each member of the gene pool that will influence the selection stage that will then steer the evolutionary process. The fitness function that we use for the parameter estimation compares the attributes of the sound output by the synthesizer running with each set of parameters inside the gene pool with attributes of the target sound.

The fitness evaluation results in a *fitness rating* that reveals the extent of similarity between two sounds. For the fitness rating, we have computed the analytical spectral distance of magnitudes between the complex spectrums is used as similarity measure of the source (synthesized) and target sound. In our implementation, we've used the analytical distance metrics proposed by [5] disregarding the phase information and focusing only on the magnitudes. The simplification of the fitness function decreases the convergence time dramatically while it preserves the perceptual quality of the synthesized sound, as it rather depends on the given synthesizer topology.

$$MSEMag = \frac{1}{Frames}\sum_{j=1}^{Frames}\sum_{i=1}^{Bins}\left[\left(|X_{ij}| - |T_{ij}|\right)^2 Wm_{ij}\right] \qquad (1)$$

$$Wm_{ij} = O + (1-O)\frac{\log|T_{ij}| - \min\left(\log|T_{ij}|\right)}{\left|\min\left(\log|T_{ij}|\right) - \max\left(\log|T_{ij}|\right)\right|} \qquad (2)$$

The Eq. 1 calculates the mean squared error (MSE) between the magnitude spectrograms of synthesized and target sounds. The weight matrix WM_{ij} (Eq. 2) helps for curtailing the errors at spectral regions with more energy. The influence of the weight matrix in MSE calculation can be adjusted with O.

5 Parallelization

The fitness evaluation is parallelizable because calculations for determining the fitness scores for all members of the gene pool in a generation can be run concurrently as the fitness function only needs the chromosome of a single individual in order to work. In our work, the evaluation of the entire population is divided between servers that are registered to the instance of the parameter estimation. Thus, the workload is efficiently partitioned for independent processing across available logical processors in a single system, and computers available in a network. Thus, it handles the synchronization and the distribution of the load across registered servers automatically. The parallelization of the parameter matching enables users to work with sound at interactive speed that is suitable for digital music performances. For that reason, we have used the client-server architecture (referred as *scsynth*) of the SC that enables parallelization of the computational tasks regarding real-time sound synthesis and analysis across the CPU cores of a single machine as well as multiple networked computers.

6 Experiments

In order to determine the efficiency of the framework, we have conducted various tests in typical usage scenarios including a percussion synthesizer and multiple modulator frequency modulation topography. All of the experiments are processed using a mobile “Intel Core 2 Duo 2.4 Ghz (Penryn)” processor by utilizing both cores.

6.1 Percussion Synthesizer (PS)

To create electronic drum and percussion sounds, we have implemented a percussion synthesizer (PS) (Fig 3) that is composed of a sine oscillator with a frequency envelope acting as a simple membrane, and a filtered noise generator envelope for helping cymbal sounds or attack noises. The sum of both envelopes is fed back to the phase input of the sine oscillator (feedback PM) and it reaches

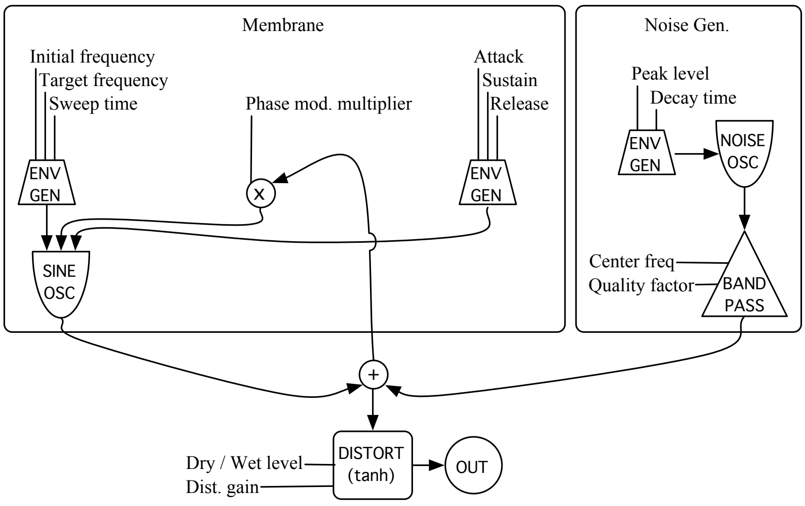


Fig. 3. Topology of the PS that creates electronic drum and percussion sounds

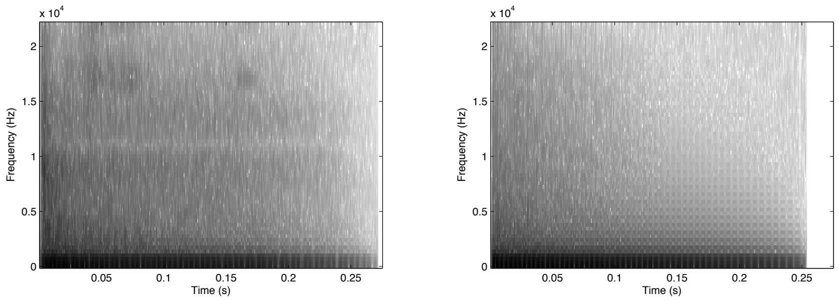


Fig. 4. Linear frequency scale spectrogram for an acoustic tom drum sample, and the output of the parameter matched by the percussion synthesizer

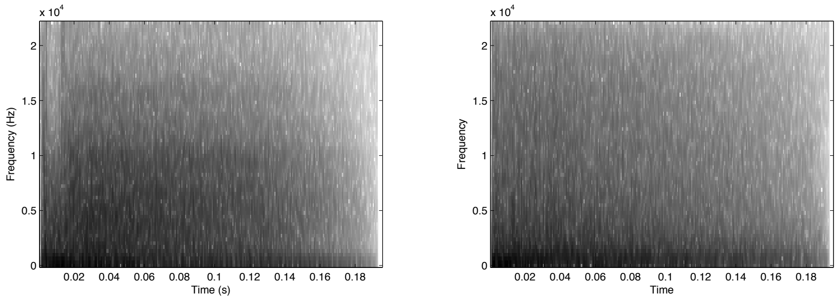


Fig. 5. Linear frequency scale spectrogram output for an acoustic snare drum sample, and the output of the parameter matched by the percussion synthesizer

the output after a distortion stage (*tanh* distortion). Almost all stages of the PS (including envelope arguments, PM amount, distortion amount, filter settings) are parameterized.

Although the PS is capable of producing a variety of electronic drum and percussion sounds, the parameter search space is not very complex for relevant sounds (such as percussive sounds that are possibly created with other drum synthesizers). However, sounds with acoustic origins have more complex spectrums; thus, PS would not be able to produce perfectly their distinctively complex spectra. Yet, we have observed that PS is capable of making cross-genre translations between sounds using our architecture. For example, PS was capable of converging to characteristic cymbal sounds that are similar to ones typically programmed for a drum synthesizer when it is fed with actual cymbal recordings for translation of that sound to an electronic version. Spectrogram views of two target (acoustic tom and snare drum samples) and synthesized sound pairs are provided in Fig. 4 and 5, where the GA converges to a fit solution in approximately 15 seconds.

6.2 Multiple Modulator Frequency Modulation (Parallel MM-FM)

A parallel MM-FM topography (Fig. 6), where 3 sine oscillators are modulating a single carrier oscillator, is implemented to test the framework in more complex search spaces. Base frequencies, modulation amounts and indexes are parametrized and no integer multiple relationships in carrier and modulator

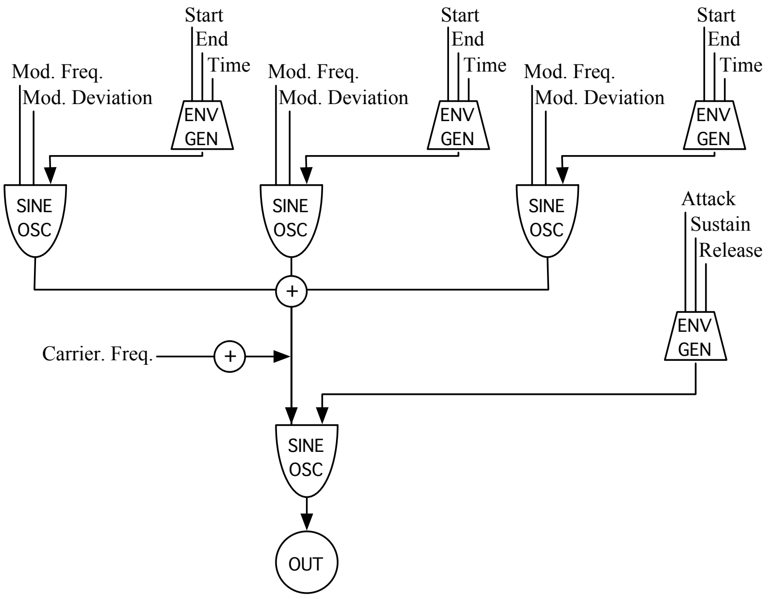


Fig. 6. Topology of the parallel MM-FM synthesizer with 3 modulators

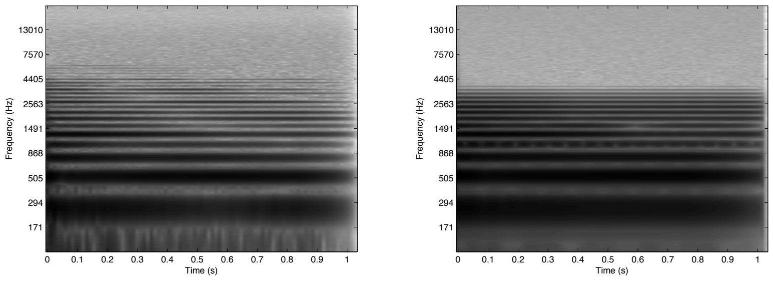


Fig. 7. Logarithmic frequency scale spectrogram output for a recorded piano sample and the output of the parameter matched synthesis by the MM-FM synthesizer

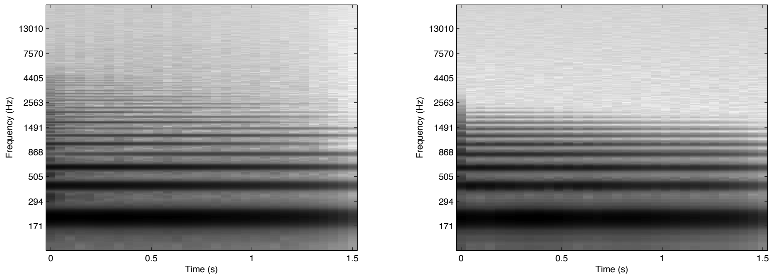


Fig. 8. Logarithmic frequency scale spectrogram output for a recorded rhodes keyboard sample and the output of the parameter matched synthesis by the MM-FM synthesizer

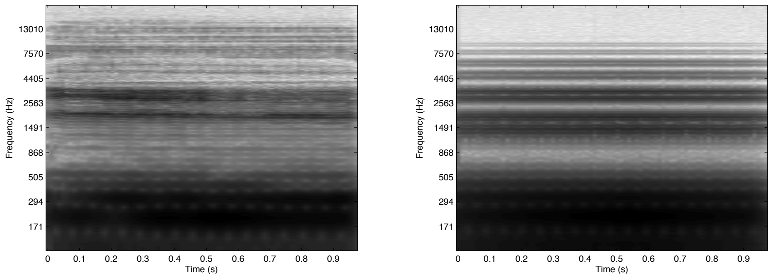


Fig. 9. Logarithmic frequency scale spectrogram output for a recorded human voice producing the "ee" vowel and the output of the parameter matched synthesis by the MM-FM synthesizer

frequencies is implied in the design. The MM-FM synthesizer is capable of producing variety of complex harmonic and inharmonic spectra. Various harmonic target sounds have been experimented using variety of GA settings; yet, the average convergence time is considerably higher (around 3 minutes) than PS because the search space is significantly more complex. When C:M ratio is locked to integer multiple relationships (without frequency envelopes), convergence time decreases substantially; however, the produced sounds become much less interesting. Spectrogram views for various target (including piano, rhodes keyboard and human voice samples) and synthesized sound pairs are provided in Figures 7, 8 and 9 respectively.

7 Discussion

The implementation was intended to be apposite for general-purpose search optimization tasks; thus, the parameter estimation system does not rely on a particular synthesis technique or synthesizer topology to be functional. Hence, the complexity of the search space is influenced directly by parameter ranges defined for the synthesizer in relation with the target sound. Paying special attention to providing possible parameter ranges is not strictly necessary, as the GA search will eliminate unintelligible parameters for a given topology. However, directing the algorithm to a set of possibly related parameter ranges would greatly decrease the complexity of the search; thus, better results might be obtained in a fixed time window. Using such setup, it might be possible to compute a parameter setting, which yields good results with virtually any sound produced by an instrument, given a few sound samples of it.

However in practical terms, there is no guarantee for an arbitrary provided method to approximate a target sound satisfactorily [16, pg 596] and usually the GA alone may not able to eliminate sets of parameters that do not represent feasible solutions. Fortunately, experimenting small or large scale synthesizer topologies is relatively easy thanks to unit generators provided within the SC distribution. Thus, synthesizers are available for sound generation immediately after defining the unit generator graph function. The process is so robust that live programming and control of synthesizers are handled real-time in live coding performances by various computer musicians [3]. The simplicity in dynamic, interactive and almost improvisatory synthesizer creation yields to some interesting opportunities like easy experimentation with cross-genre transformation of sounds and errant approaches to parameter estimation between seemingly unfit synthesizers and targets for creative purposes.

8 Conclusion

In this work, we have proposed a general-purpose evolutionary framework, which is integrated into the SuperCollider (SC) software platform, to perform flexible parameter estimation for digital sound synthesizers. The system is able to provide solutions for parameter estimation tasks at interactive speeds typically necessitated for live performance and composition workflows. The modular and flexible structure of the framework may open wide-range of opportunities for musicians and researchers in the field.

References

1. Boulanger, R.C.: The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming. MIT Press, Cambridge (2000)
2. Cheung, N.M., Horner, A.B.: Group Synthesis with Genetic Algorithms. Journal of the Audio Engineering Society 44(3) (1996)
3. Collins, N., McLean, A., Rohrhuber, J., Ward, A.: Live Coding in Laptop Performance. Organised Sound 8(3), 321–330 (2003)
4. Fornari, J.: An ESSynth Implementation in PD. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Research Posters, p. 106. ACM, New York (2006)
5. Garcia, R.: Growing Sound Synthesizers Using Evolutionary Methods. In: European Conference in Artificial Life ECAL 2001. Artificial Life Models for Musical Applications, University of Economics, Prague (2001)
6. Horner, A.: Wavetable Matching Synthesis of Dynamic Instruments with Genetic Algorithms. Journal of Audio Engineering Society 43(11), 916–931 (1995)
7. Horner, A.: Nested Modulator and Feedback FM Matching of Instrument Tones. IEEE Transactions on Speech and Audio Processing 6(4) (1998)
8. Horner, A.: Double-Modulator FM Matching of Instrument Tones. Computer Music Journal 20(2), 57–71 (1996)
9. Horner, A., Beauchamp, J., Haken, L.: Machine Tongues XVI: Genetic Algorithms and Their Application to FM Matching Synthesis. Computer Music Journal 17(4), 17–29 (1993)
10. Johnson, C.: Exploring Sound-Space with Interactive Genetic Algorithms. Leonardo 36(1), 51–54 (2003)
11. Manzolli, J., Maia Jr., A., Fornari, J., Damiani, F.: The Evolutionary Sound Synthesis Method. In: MULTIMEDIA 2001: Proceedings of The Ninth ACM International Conference on Multimedia, pp. 585–587. ACM, New York (2001)
12. McCartney, J.: SuperCollider, a New Real Time Synthesis Language. In: Proceedings of the International Computer Music Conference, pp. 257–258 (1996)
13. McCartney, J.: Rethinking the Computer Music Language: SuperCollider. Computer Music Journal 26(4), 61–68 (2002)
14. McDermott, J., O'Neill, M., Griffith, J.L.: Target-driven Genetic Algorithms for Synthesizer Control. In: 9th Int. Conference on Digital Audio Effects, DAFx 2006 (2006)
15. Miller, B.L., Goldberg, D.E.: Genetic Algorithms, Tournament Selection, and the Effects of Noise. Complex Systems 9(3) (1995)
16. Roads, C., Strawn, J.: The Computer Music Tutorial, 4th edn. MIT Press, Cambridge (1999)
17. Wright, M.: Open Sound Control: An Enabling Technology for Musical Networking. Organised Sound 10(3), 193–200 (2005)

Combining Structural Analysis and Multi-Objective Criteria for Evolutionary Architectural Design

Jonathan Byrne, Michael Fenton, Erik Hemberg, James McDermott, Michael O'Neill, Elizabeth Shotton, and Ciaran Nally

Natural Computing Research & Applications Group
University College Dublin, Ireland
{jonathanbyrn,michaelfenton1}@gmail.com,
m.oneill@ucd.ie

Abstract. This study evolves and categorises a population of conceptual designs by their ability to handle physical constraints. The design process involves a trade-off between form and function. The aesthetic considerations of the designer are constrained by physical considerations and material cost. In previous work, we developed a design grammar capable of evolving aesthetically pleasing designs through the use of an interactive evolutionary algorithm. This work implements a fitness function capable of applying engineering objectives to automatically evaluate designs and, in turn, reduce the search space that is presented to the user.

1 Introduction

Design can be described as a purposeful yet explorative activity [7]. Initially the designer must explore the search space to find a concept or form that is capable of fulfilling the design specification. Once the form has been chosen, the design process focuses on satisfying the constraints of the original design specification. At the centre of this process there is a conflict between form and function. While these two attributes of a design are not mutually exclusive, there can be a trade off when realising a design.

In this paper we look at the specific case of architectural design. For a structure to be created it requires the combined effort of both architects and engineers. The heuristics an architect uses to evaluate a design are not the same as a structural engineer. Architects evaluate all aspects of the design, from broader issues of internal and external relationships to more detailed aesthetic measures such as material use, texture and light. Engineers evaluate the integrity of the structure itself. To oversimplify, architects are concerned with spaces, engineers are concerned with forces.

This study is a continuation of our previous work that primarily focused on the aesthetic qualities of conceptual design [19]. Through the use of design grammars and an interactive fitness function we have shown that Grammatical Evolution (GE) [18] is capable of creating surprising and innovative designs. Conversely,

© Springer-Verlag Berlin Heidelberg 2011

the focus of structural evolutionary design has primarily been concerned with engineering constraints. The objective nature of engineering constraints lend themselves to implementation as a fitness function and allow a design to be optimised accordingly [15].

Our work combines the formal approach of architecture with the constraints of engineering. The advantages of this approach are twofold. First, conceptual designs can be optimised to increase their functionality and strength while reducing the amount of material used. This will, in turn, make the designs more credible and realisable. Second, assigning an objective fitness to a design also provides a mechanism for grouping designs. The user may then select which areas of the search space they find the most interesting and thus accelerate convergence on aesthetically pleasing designs.

This paper is organised as follows. Section 2 is a summary of related research in this area. A description of our approach to design generation and analysis is given in Section 3. The two experiments carried out using this system are described in Section 4 and Section 5 and their results are examined. Our conclusions and future work are discussed in Section 6.

2 Previous Work

Computers are ubiquitous in design but they are typically used as an analytical aid rather than as a generative tool. Computer applications are employed after the conceptual design process has been completed. With a few notable exceptions, the computer is not used to explore the search space of possible designs. This section discusses previous work in design generation.

2.1 Conceptual Evolutionary Design

A direct approach that allows the designer to explore the design search space is to implement a parametric system. The user inputs their design and then modifies individual components of that design. EIFForm was a successful attempt at implementing parametric design and the results have been used to design a structure in the inner courtyard of Schindler house [22]. Parametric design tools have now been introduced into more mainstream design software. There is the Grasshopper plug-in for the Rhino modelling system [9] and Bentley Systems have implemented a program called Generative Components [6].

An evolutionary approach to conceptual design exploration is implemented in GENR8 [20]. This system uses GE and Hemberg Extended Map L-Systems (HEMLS) to generate forms. The user can influence the growth of the L-System through the use of tropism and fitness weighting. Objects can be placed in the environment that either attract or repel the design. Each design is evaluated to a series of metrics, symmetry, undulation, size smoothness, etc. The user is able to weight these metrics according to their preference. Our approach builds on this work. We use a grammar for generating designs and a combination of automatic and users evaluation to drive the evolutionary process.

2.2 Evolutionary Structural Design

Structural engineers seek to find ways for a structure to resist the applied stresses while also reducing material usage and cost. Evolutionary Computation (EC) naturally lends itself to these problems and, accordingly, there has been a large amount of work in this area. Many of the earliest EC applications were focused on optimising structures [15]. The computational cost of structural analysis meant that early papers focused on greatly simplified structures, such as two dimensional trusses [11]. As computational power increased, so did the scope of the applications. Structures such as bridges [25], electricity pylons [23], and even whole buildings [14] have been optimised using EC. A list of applications is covered extensively in the literature of Kicinger [15]. Structural optimisation falls into three categories. The overall layout of the system (topology), the optimal contour for a fixed topology (shape) and the size and dimensions of the components (sizing). Our work focuses on the topological optimisation, although the modular nature of our approach could be adapted for optimising the other categories. This possibility is discussed in greater detail in Section 6.

2.3 Interactive Evolutionary Computation

Interactive Evolutionary Computation (IEC) was developed as a means of assigning fitness when no objective metric could be defined. Human interaction has allowed EC to be applied to problems such as music and computer graphics, and to act as an exploratory tool as opposed to its primary function as an optimiser. A more complete list of interactive applications can be found in [1] and [24].

3 Experimental Setup

Our system is comprised of four parts, an evolutionary algorithm, a design grammar, structural analysis software and a multi-objective fitness function. This section describes our approach to generating and evaluating designs.

3.1 Grammatical Evolution

Grammatical Evolution is an evolutionary algorithm that is based on GP [18]. It differs from standard GP by representing the parse-tree based structure of GP as a linear genome. It accomplishes this by using a Genotype-Phenotype mapping of a chromosome represented by a variable length bit or integer string. The chromosome is made up of codons eg:(integer based blocks). Each codon in the string is used to select a production rule from a Backus Naur Form(BNF) grammar. Production rules are selected from the grammar until all non-terminal rules are mapped and a complete program is generated. The advantage of using a grammar is that it is possible to generate anything that can be described as a set of rules. Grammars are capable of generating strings, mathematical formulas, pieces of programming code and even whole programs. The grammar used for our experiments is described in Section 3.2. Another advantage of applying

GE to design is that generative processes, like the mapping process in GE, are required for the production of designs that can scale to a high level of complexity [13].

3.2 Design Grammar

The grammar was originally conceived based on a brief provided to third year students in the UCD architecture and structural engineering course of 2010. The brief specified that the bridge was to be composed of timber, had an optional arch, a width of 2 metres and bridge a span of 10 metres. In our previous experiment, evaluation was provided solely from user interaction. The grammar was created with no consideration for the structural soundness of the resulting bridges. Despite this, it was possible to compare the relative performance of bridges in the grammar by applying a pre-determined loading.

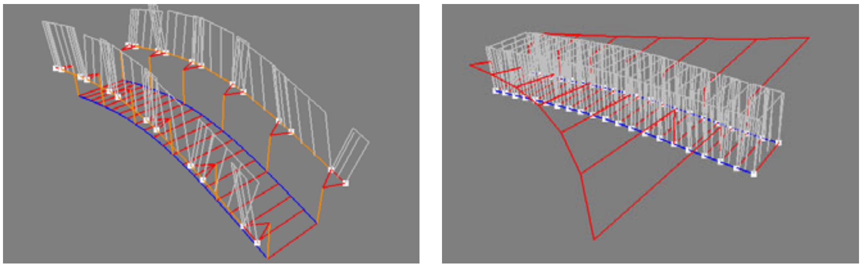
The size of the grammar meant that it could not be appended to the paper. The grammar is available online at [16]. The grammar creates graphs using networkx [10], a python class for studying complex graphs and networks. Three desirable characteristics for a design generator are modularity, regularity and hierarchy [12]. We implement these characteristics using the novel method of higher order functions. Our work in this area is discussed in greater detail in [17]. For structural analysis to be performed on the bridges, a mechanism was required for specifying the loads on the structure. Our approach was to add attributes to the existing grammar. This allowed us to label components depending on the function that created them. Labelling meant that forces could be assigned to the structures automatically and accordingly, that different forces could be applied to different parts of the bridge. An example of this can be seen in Figure 1.

3.3 Structural Analysis

The ability to analyse structures as computable models is achieved by using Finite Element Methods [8]. Instead of calculating the partial differential equation for a whole design, a continuous structure is discretised into an approximating system of ordinary differential equations. The approximation can then be solved using numerical approximation methods for differentiation such as Euler's method or the Rung-Kutta method. Our designs are particularly suited to Finite Element Analysis (FEA) as the structures are already discretised into a series of interconnected beams. To analyse our designs we are using San Le's Free Finite Element Analysis (SLFFEA) [21]. This software is freely available for download and has been used by engineering companies in industry.

3.4 Multi-Objective Fitness Function

Design usually involves satisfying several (possibly contradictory) objectives. Multi-objective evolutionary algorithms (MOEAs) have been shown to be a useful approach for finding the best compromise when tackling a multi-objective problem [26]. Instead of weighting the objectives and allowing an evolutionary



(a) Force being applied to the handrail (b) Force being applied to the walkway

Fig. 1. Different Magnitudes of Stresses Being Applied to the Handrail and Walkway

algorithm to converge on a single global optimum, the algorithm builds a pareto-front of the individuals that maximise the given objectives. Using fronts can aid the design process by presenting the user with several pareto-equivalent designs and letting them select the design that most closely matches their requirements. We are using a GE implementation of the NSGA2 algorithm [3] as our selection mechanism. The algorithm uses a fast non-dominated sorting algorithm to calculate which individuals are on the front and then group the other individuals in the population relative to this group. Normally MOEA applications are only concerned with the individuals on the pareto-front, we intend to investigate in Section 5 whether the grouping property of the NSGA2 algorithm could also be of benefit for guiding the search process.

4 Optimising Designs Using Structural Analysis

This experiment aimed to test whether an evolutionary search was capable of generating designs that minimised the stress in a structure and reduced the amount of material used. It was carried out using the implementation described in Section 3 and the bridge grammar described in Section 3.2. The experimental settings were: Population size = 100, Generations = 50, No. of Runs = 30, Mutation Rate = 1.5%, Crossover Rate = 70%, Selection Scheme = Tournament, Tournament Size = 3, Replacement Scheme = NSGA2.

The material from which the bridge was constructed was small scale air dried oak sections with a moisture content of 20% or more. The structural qualities of this wood were taken from the British Standards BS-EN-338-2003 as a grade D30 class of timber [2]. The material qualities were then assigned to the bridge beams for SLFFEA analysis. For stresses on a structure to be calculated, we must first assign fixed points and loaded beams. Normally this is done manually by the user. Our approach automated this by using attributes in the grammar, as described in Section 3.2. The bridges were subjected to a uniformly distributed load (UDL) of 5kN/m upon the walkway itself and a separate 1kN/m load was applied to the handrails. The loads for the bridge were taken from [4].

While we tried to replicate a load that a bridge might be subjected to during actual usage, the main purpose was to compare how well the bridges performed relative to other bridges generated by the design grammar.

There were two constraints placed on the designs, one of which was stress based and one that was based on material usage. The stress constraint in the fitness function calculated the maximum stress on each beam in the design, this was then averaged over the whole structure and the selection pressure aimed at reducing it. If a beam failed then the bridge was assigned a default fitness of 100,000. This meant that high stress designs were removed from the population and the fitness pressure aimed at reduced stresses over the structure as a whole. The material constraint aimed at reducing the number of beams used in a structure. This fitness metric is opposed to the stress constraint as one method for reducing the average stress on the beams is by adding more unnecessary beams. By adding a penalty for the total weight of material used, it can force the algorithm to simplify the design. Reducing the total weight of material used also translates into direct savings when manufacturing an instance of the design.

4.1 Optimisation Results

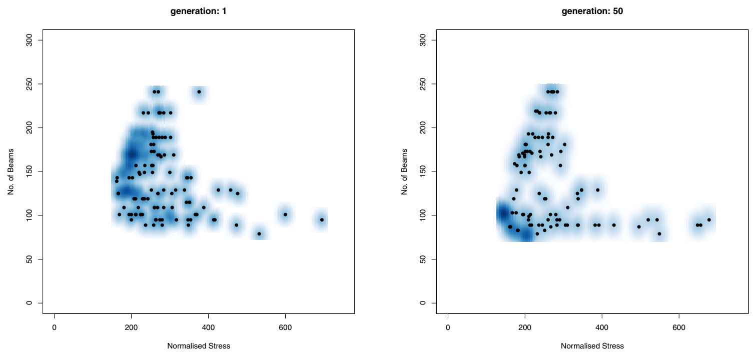
The results for the experiment are shown in Figures 2 and 3. It is clear that the fronts are moving toward a pareto-optimality over the course of 50 generations, as shown in Figure 2. There is a 43% reduction in the material used (Figure 3(a)) and a reduction of the average maximum stress placed on the structure of 41% (Figure 3(b)) after 50 generations. The results show that using structural analysis and an MOEA can significantly reduce the stresses and self weight of a design.

The results show that our system is capable of evolving structures that increasingly satisfy the constraints specified in our multi-objective fitness function. This is very important for trying to move a design from a mere concept to something that could be actualised. This is a challenge that faces engineers and architects on a daily basis and a GE based approach such as this has the potential to help solve this problem.

5 Categorising Designs Using Structural Analysis

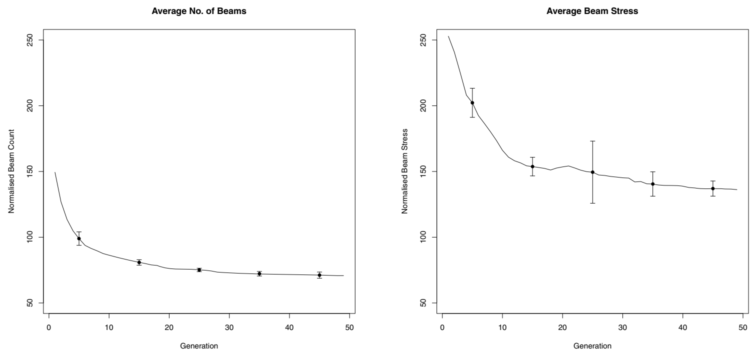
Our intention in implementing this software is not to exclusively optimise designs but to allow the architect to interactively evolve designs that they find aesthetically pleasing. To this end, it is imperative to bias the algorithm towards designs that the user finds interesting and appealing. The design process is not about optimisation and, as such, designers are often interested in designs that do not lie on the pareto front.

In this experiment we used the settings described previously except that we only allowed each run to be executed for a single generation. Instead of using the NSGA2 algorithm to optimise the bridge designs, it is used to group the bridge designs by realisability. The grouping created by the fast non-dominated sort are shown in different colors in Figure 5. The user selects the design grouping they



(a) Generation 1 (b) Generation 50

Fig. 2. Scatter plot with a density estimation function that shows the progression of front over 50 generations



(a) Normalised Average of Beams Used (b) Normalised Average of Stress

Fig. 3. The fitness minimisation average of the population over 50 generations

find the most interesting and so direct the pareto-front using selection pressure. By categorising designs by their place on the fitness landscape we can accelerate convergence onto more appealing areas of the search space.

In our experiment, we randomly selected designs from the first two fronts and the last two non-empty fronts.To generate images of the designs we used an open source mesh viewer developed by the INRIA called medit [5]. An online survey was then conducted on the designs. The survey consisted of presenting two designs,side by side, and asking the user to select which design they found most aesthetically pleasing, as shown in Figure 4. If you wish to see more of the designs, there is a link to the survey at [16]. If the user had no preference for a particular design they can indicate this with the no preference button. The presentation of the images were randomised so that there was no bias for which

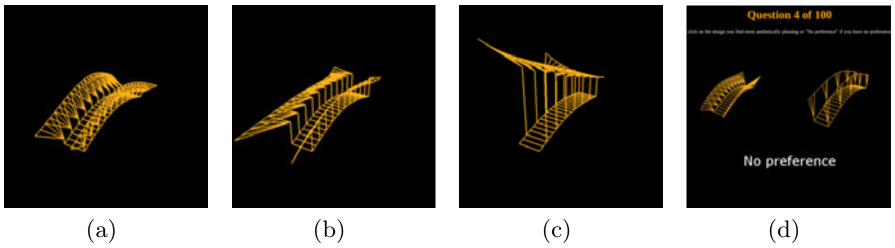


Fig. 4. Sample bridges from the survey and survey layout (d)

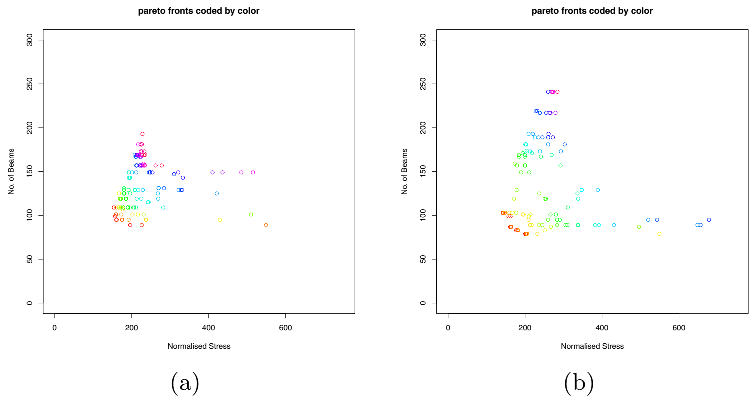


Fig. 5. Color differentiated fronts produced by the NSGA2 fast non-dominated sorting algorithm at generation 1 (a) and generation 50 (b)

side the images appear on. The survey was carried out by post-graduate students and volunteers from the school of architecture. This survey was authorised by the ethics committee and the head of the school of Architecture.

5.1 Categorisation Results

The survey was completed by 28 individuals and consisted of 2800 evaluations. The users showed a preference of 55.9% for bridges from the end of the non-dominated sort compared to a 36.84% preference for non-dominated bridges from the front of the sort. The users had no preference on 7.26% of the designs. This shows an aesthetic preference for designs that do not fulfill the engineering constraints. The results imply that the engineering constraints that we chose for this experiment are in opposition to aesthetic preferences. Although aesthetic preference is a purely subjective quality, the inclination towards unconstrained designs could be because of their unusual and unexpected configurations rather than the “ordinary” nature of structurally sound designs.What is most interesting about this result was that there was no selection pressure on the population. The fast non-dominating sort was applied to randomly generated individuals.

As can be seen in Figure 5(a) compared to 5(b), this is the worst case scenario. If a population was evolved over several generations the difference between individuals greatly increases, which would aid categorisation of the population.

The results indicate that the fast non-dominated sort in the NSGA2 algorithm has a secondary purpose; It can be used to group designs by how well they meet the objectives. This mechanism could greatly speed up IEC by allowing the user to choose between groups rather than selecting individuals.

6 Conclusion and Future Work

In this paper we encoded material and physical constraints into a fitness function and showed conceptual designs could be evolved towards those objectives. This is step towards making conceptual designs more realisable. We also showed that multi-objective fitness functions could be used for more than optimisation. By automatically categorising the designs and then presenting those categories to a user for evaluation, the MOEA could drastically reduce the search space presented to the user during IEC.

Our future work intends to encode other aesthetic constraints such as smoothness, curvature, etc and allow the user to select objectives that they would most like to see in the presented design. The modular structure of our software makes it possible to lock the topology of a chosen bridge design and focus on the optimisation of either the shape of the beams or the sizing and material the beams are constructed from.

Acknowledgments

We would like to thank Andrea McMahon and Brian Lynch for their unceasing support while writing this paper. This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/RFP/CMS1115 and the Graduate Research Education Programme in Sustainable Development, jointly funded by IRCSET and IRCHSS.

References

1. Banzhaf, W.: Interactive evolution. In: Back, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation, ch. C2.9, pp. 1–6. IOP Publishing Ltd., Oxford University Press (1997)
2. British Standards Institution: BS EN 338-2003: Structural Timber Strength Classes. BSI, London (2003)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Fenton, M.: Analysis of Timber Structures Created Using A G.E-Based Architectural Design Tool. Master's thesis, University College Dublin, Ireland (2010)
5. Frey, P.J.: MEDIT:interactive mesh visualization. 0 RT-0253, INRIA (December 2001), http://hal.inria.fr/inria-00069921/en/

6. Generative Components, http://www.bentley.com/getgc/
7. Gero, J.S.: Creativity, emergence and evolution in design. Knowledge-Based Systems 9(7), 435–448 (1996)
8. Glaylord, E., Glaylord, C.: Structural engineering handbook. McGraw-Hill, New York (1979)
9. Grasshopper, Generative Modeling with Rhino, http://www.grasshopper3d.com/
10. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using networkx. In: Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, pp. 11–15 (2008)
11. Hoeffler, A., Leysner, U., Weidermann, J.: Optimization of the layout of trusses combining strategies based on Mitchels theorem and on biological principles of evolution. In: Proceeding of the 2nd Symposium on Structural Optimisation, Milan, Italy (1973)
12. Hornby, G.S.: Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In: Proceedings of GECCO 2005 (2005)
13. Hornby, G.S., Pollack, J.B.: The advantages of generative grammatical encodings for physical design. In: Proceedings of the 2001 Congress on Evolutionary Computation CEC 2001, pp. 600–607. IEEE Press, Los Alamitos (2001)
14. Kicinger, R., Arciszewski, T., DeJong, K.: Evolutionary design of steel structures in tall buildings. Journal of Computing in Civil Engineering 19(3), 223–238 (2005)
15. Kicinger, R., Arciszewski, T., Jong, K.D.: Evolutionary computation and structural design: A survey of the state-of-the-art. Computers and Structures 83(23-24), 1943–1978 (2005)
16. Link to the bridge grammar, http://i.imgur.com/0vsDh.png
17. McDermott, J., Byrne, J., Swafford, J.M., O'Neill, M., Brabazon, A.: Higher-order functions in aesthetic EC encodings. In: 2010 IEEE World Congress on Computational Intelligence, pp. 2816–2823. IEEE Press, Barcelona (2010)
18. O'Neill, M.: Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. Ph.D. thesis, University Of Limerick, Ireland (2001)
19. O'Neill, M., McDermott, J., Swafford, J.M., Byrne, J., Hemberg, E., Shotton, E., McNally, C., Brabazon, A., Hemberg, M.: Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. International Journal of Design Engineering (in press)
20. O'Reilly, U.M., Hemberg, M.: Integrating generative growth and evolutionary computation for form exploration. Genetic Programming and Evolvable Machines 8(2), 163–186 (2007), special issue on developmental systems
21. San Lee's Free Finite Element Analysis, http://slffea.sourceforge.net/
22. Shea, K., Aish, R., Gourtovaia, M.: Towards integrated performance-driven generative design tools. Automation in Construction 14(2), 253–264 (2005)
23. Shea, K., Smith, I., et al.: Improving full-scale transmission tower design through topology and shape optimization. Journal of Structural Engineering 132, 781 (2006)
24. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proc. of the IEEE 89(9), 1275–1296 (2001)
25. Topping, B., Leite, J.: Parallel genetic models for structural optimization. Engineering Optimization 31(1), 65–99 (1988)
26. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evolutionary Computation 3(4), 257–271 (1999)

Music Translation of Tertiary Protein Structure: Auditory Patterns of the Protein Folding

Riccardo Castagna[1,*], Alessandro Chiolerio[1], and Valentina Margaria[2]

[1] LATEMAR - Politecnico di Torino
Dipartimento di Scienza dei Materiali ed Ingegneria Chimica
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
Tel.: +39 011 0907381
riccardo.castagna@polito.it
[2] Independent Researcher

Abstract. We have translated genome-encoded protein sequence into musical notes and created a polyphonic harmony taking in account its tertiary structure. We did not use a diatonic musical scale to obtain a pleasant sound, focusing instead on the spatial relationship between aminoacids closely placed in the 3-dimensional protein folding. In this way, the result is a musical translation of the real morphology of the protein, that opens the challenge to bring musical harmony rules into the proteomic research field.

Keywords: Bioart, Biomusic, Protein Folding, Bioinformatics.

1 Introduction

During recent years, several approaches have been investigated to introduce biology to a wider, younger and non-technical audience [2, 10]. Accordingly, bio-inspired art (Bioart) represents an interdisciplinary field devoted to reduce the boundaries between science, intended as an absolutely rational argument, and the emotional feelings. By stimulating human senses, such as sight, touch and hearing, scientists and artists together attempt not just to communicate science but also to create new perspectives and new inspirations for scientific information analysis based on the rules of serendipity [13].

Bio-inspired music (Biomusic) is a branch of Bioart representing a well developed approach with educational and mere scientific aims. In fact, due to the affinity between genome biology and music language, lot of efforts have been dedicated to the conversion of genetic information code into musical notes to reveal new auditory patterns [4, 6, 7, 10-12, 14, 15].

The first work introducing Biomusic [10] showed the attempt to translate DNA sequences into music, converting directly the four DNA basis into four notes. The goal was initially to create an acoustic method to minimize the distress of handling the increasing amount of base sequencing data. A certain advantage of this approach

[*] Corresponding author.

© Springer-Verlag Berlin Heidelberg 2011

was that the DNA sequences were easily recognized and memorized, but from an aesthetic/artistic point of view it represented a poor result due to the lack of musicality and rhythm. Other approaches to convert DNA sequences into music were based on codons reading frame and mathematical analysis of the physical properties of each nucleotide [6, 7].

Unfortunately, because of the structure and the nature of DNA, all these attempts gave rise to note sequences lacking of musical depth. In fact, since the DNA is based on four nucleotides (Adenine, Cytosine, Guanine, Thymine), a long and un-structured repetition of just four musical notes is not enough to create a musical composition. As a natural consequence, scientists focused their attention on proteins, instead of DNA, with the aim of obtaining a reasonable, pleasant and rhythmic sound that can faithfully represent genomic information.

Proteins are polymers of twenty different amino acids that fold into specific spatial conformations, driven by non-covalent interactions, to perform their biological function. They are characterized by four distinct levels of organization: primary, secondary, tertiary and quaternary structure.

The primary structure refers to the linear sequence of the different amino acids, determined by the translation of the DNA sequence of the corresponding gene. The secondary structure, instead, refers to regular local sub-structures, named alpha helix (α-helix) and beta sheet (β-sheet). The way the α-helices and β-sheets folded into a compact globule describes the tertiary structure of the protein. The correct folding of a protein is strictly inter-connected with the external environment and is essential to execute the molecular function (see Figure 1). Finally, the quaternary structure represents a larger assembly of several protein molecules or polypeptide chains [3].

A number of studies have dealt with the musical translation of pure protein sequences [4, 15]. For example, Dunn and Clark used algorithms and secondary structure of proteins to translate amino acid sequences into musical themes [4].

Another example of protein conversion in music is given by the approach used by Takahashi and Miller [15]. They translated the primary protein structure in a sequence of notes, and after that they expressed each note as a chord of a diatonic scale.

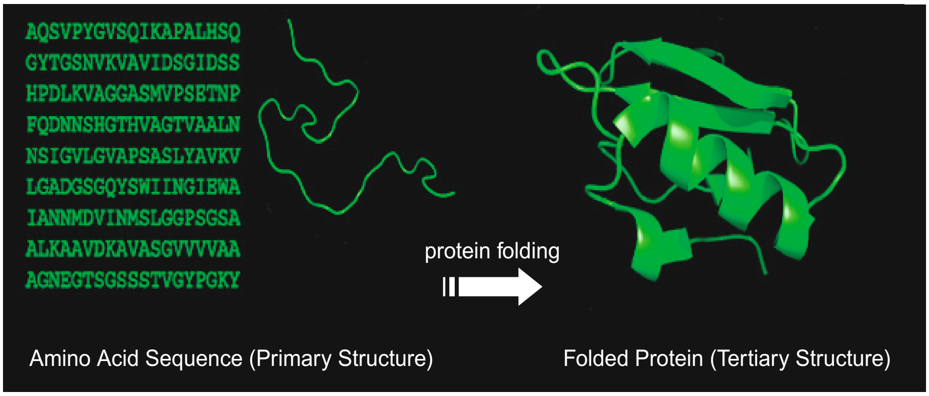


Fig. 1. Protein structures: linear sequence of amino acids, described on the left, fold into specific spatial conformations, driven by non-covalent interactions

Moreover, they introduced rhythm into the composition by analyzing the abundance of a specific codon into the corresponding organism and relating this information with note duration. Anyway, the use of the diatonic scale and the trick of chords built on a single note gave rise to results that are partially able to satisfy the listener from a musical point of view but, unfortunately, they are not a reliable representation of the complexity of the molecular organization of the protein.

Music is not a mere linear sequence of notes. Our minds perceive pieces of music on a level far higher than that. We chunk notes into phrases, phrases into melodies, melodies into movements, and movements into full pieces. Similarly, proteins only make sense when they act as chunked units. Although a primary structure carries all the information for the tertiary structure to be created, it still "feels" like less, for its potential is only realized when the tertiary structure is actually physically created [11]. Consequently, a successful approach for the musical interpretation of protein complexity must take in account, at least, its tertiary structures and could not be based only on its primary or secondary structure.

2 Method

Our pilot study focused on the amino acid sequence of chain A of the Human Thymidylate Synthase A (ThyA), to create a comparison with the most recent work published on this subject [15]. The translation of amino acids into musical notes was based on the use of Bio2Midi software [8], by means of a chromatic scale to avoid any kind of filter on the result.

The protein 3-dimensional (3D) crystallographic structure was obtained from the Protein Data Bank (PDB). Information relative to the spatial position in a 3D volume of each atom composing the amino acids was recovered from the PDB textual structure (http://www.rcsb.org/pdb/explore/explore.do?structureId=1HVY).

The above mentioned file was loaded in a Matlab® environment, together with other useful indicators such as the nature of the atom and its sequential position in the chain, the corresponding amino acid type and its sequential position in the protein.

A Matlab® script was written and implemented to translate the structure in music, as described below. We adopted an approach based on the computation of the centre of mass of each amino acid, which was identified and used as the basis for subsequent operations: this was done considering each atom composing every amino acid, its position in a 3D space and its mass. Therefore, the mass-weighed mean position of the atoms represented the amino acid centre of mass.

3 Results

3.1 Distance Matrix: Musical Chords

The first important output obtained from the algorithm was the distance matrix, containing the lengths of the vectors connecting the amino acids one by one. The matrix is symmetrical by definition and features an interesting block structure (see Figure 2). The symmetry is explained simply considering that the distance

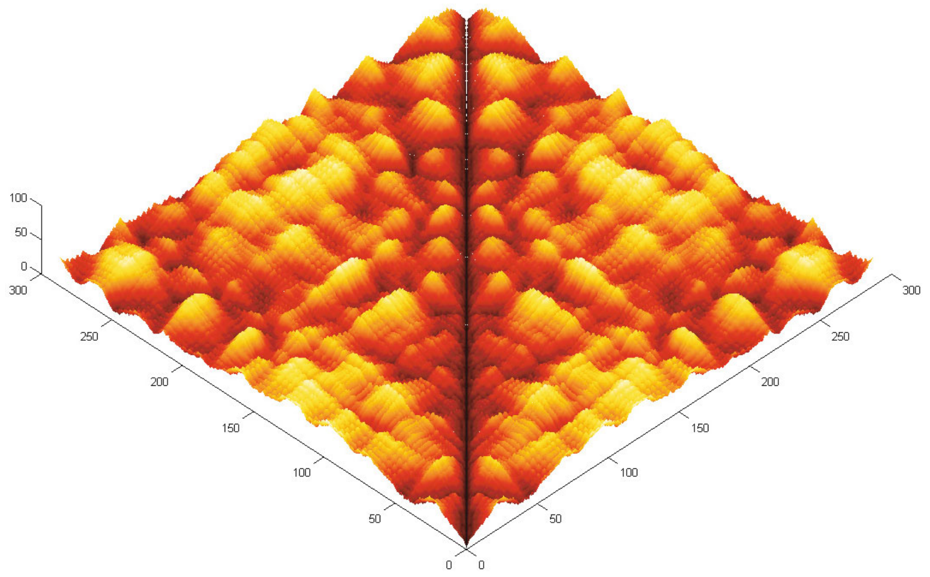


Fig. 2. Distance matrix. X and Y axis: sequential number of amino acid; Z axis: distance in pm (both vertical scale and colour scale).

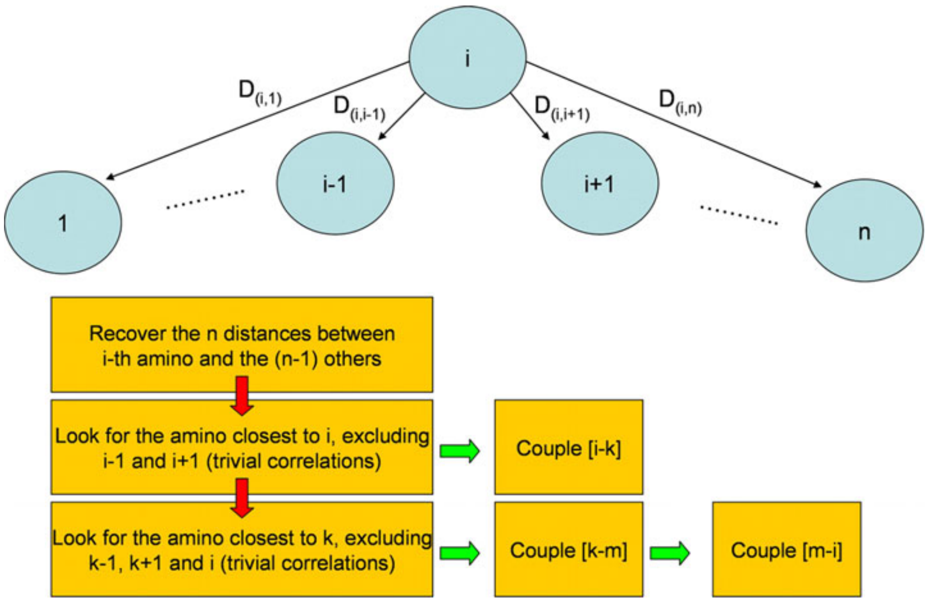

i
$D_{(i,1)}$
$D_{(i,i-1)}$
$D_{(i,i+1)}$
$D_{(i,n)}$
1
i-1
i+1
n
Recover the n distances between i-th amino and the (n-1) others
Look for the amino closest to i, excluding i-1 and i+1 (trivial correlations)
Couple [i-k]
Look for the amino closest to k, excluding k-1, k+1 and i (trivial correlations)
Couple [k-m]
Couple [m-i]

Fig. 3. Sketch of the Matlab® script operation

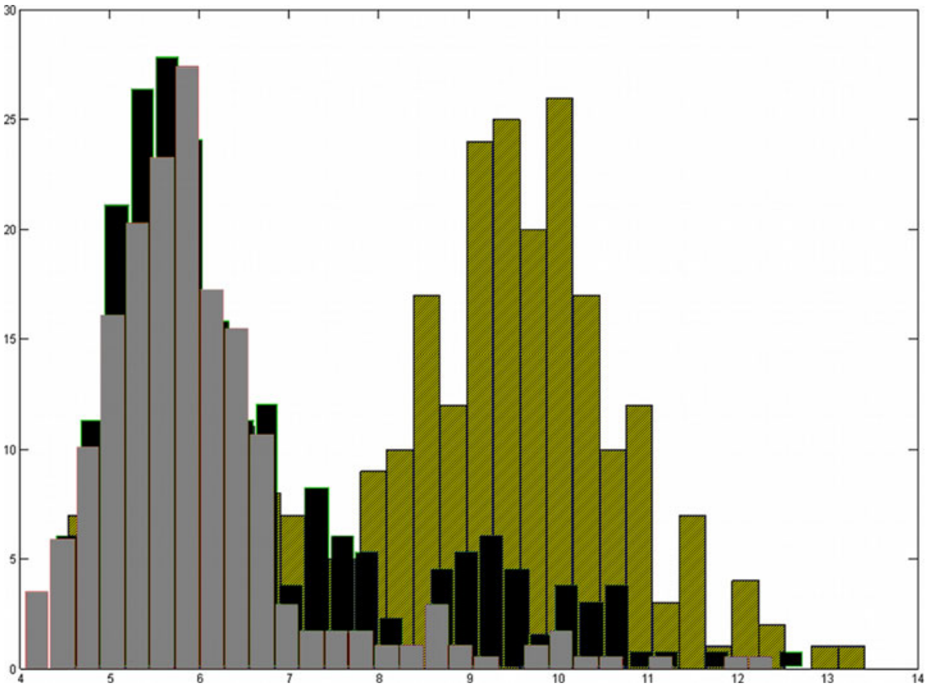


Fig. 4. Distance distributions. Histograms depicting the distance distribution in the three orders(X axis: distance in pm; Y axis: number of amino acid couples). Selection rules avoid the nearest-neighbours amino acids. Increasing the cut-off to third order it is possible to sample the second peak of the bi-modal distribution (diagonal lines).

between the i-th and j-th amino acid is the same of that between the j-th amino acid and the i-th one. The block structure is probably due to amino acid clustering in portions of the primary chain.

We sought spatial correlations between amino acids as non-nearest-neighbour, hence ignoring those amino acids which are placed close one to the other along the primary sequence. By spatial correlation, we mean the closest distance between non-obviously linked amino acids.

Running the sequence, the Matlab® script looked for three spatial correlations (i.e. three minimal distances) involving three different amino acids (two by two), as sketched in Figure 3. The two couples for every position in the primary chain were then stored and the corresponding distance distribution was extracted and plotted (see Figure 4). Those spatial correlations, or distances, are addressed to as first, second and third order. The parallelism between the sequence order and the discretization order ubiquitous in every field of digital information theory emerges from the spatial description of the protein: the more precise is the observation of the amino acids in proximity of a certain point, the higher the order necessary to include every spatial relation. The same concept applies to digital music: the higher either the bit-rate (as in MP3 codification) or the sampling frequency (as in CD-DA), the higher the fidelity. The topmost limit is an order equal to n, the number of amino acids composing a protein.

3.2 Note Intensity: Musical Dynamics

The note intensity is given in a range between 0 and 99. We assumed that the closer is the couple of amino acids, the higher is the intensity of the musical note. In order to play each order with an intensity comparable to the first order, characterized by the closest couples which may be found in the whole protein structure, we performed a normalization of the distance data within each order. In this way, normalized distance data, multiplied times 99, give the correct intensity scale.

3.3 Angle Distribution: Musical Rhythm

The primary sequence was analyzed also to extrapolate the degree of folding, a measure of the local angle between segments ideally connecting the centres of mass of subsequent amino acids.

Proteins composed by extended planar portions β-sheet tend to have an angular distribution centred around 180°. The angle distribution was extracted (see Figure 5) and parameterized as the note length: the more linear is the chain, the shorter is the note. This step gave rhythm to the generated music. In this way, the musical rhythm is intended as the velocity of an imaginary visitor running on the primary sequence. We would like to point out that this conversion features a third order cut-off, meaning that the spatial description fidelity is based on three spatial relations for each amino acid position; the higher is the cut-off, the higher the sound quality.

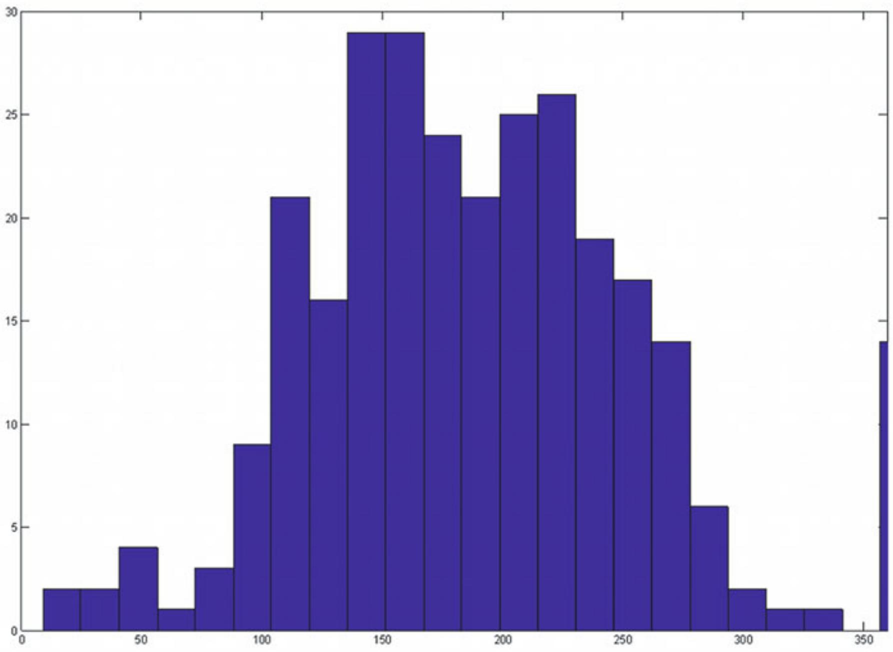


Fig. 5. Angular distribution. Histogram showing the angular distribution of the vectors linking the amino acids along the chain A of the human Thymidylate Synthase A (ThyA), used to codify each note length.

3.4 Output: Musical Score/Notation

Finally, a conversion from each amino acid to the corresponding note was performed, generating an ASCII textual output that can be converted to a MIDI file with the GNMidi software [9].

Since amino acids' properties influence the protein folding process, we adopted Dunn's translation method [4] that is based on amino acids water solubility. The most insoluble residues were assigned pitches in the lowest octave, the most soluble, including the charged residues, were in the highest octave, and the moderately insoluble residues were given the middle range. Thus, pitches ranged over two octaves for a chromatic scale.

After that, the MIDI files of the three orders were loaded in Ableton Live software [1] and assigned to MIDI instruments. We chose to assign the first order sequence of musical notes to a lead instrument (Glockenspiel) and to use a string emulator to play the other two sequences (see Figure 6). In this way it is possible to discern the primary sequence from the related musical texture that represents the amino acids involved in the 3D structure (See Additional Data File).

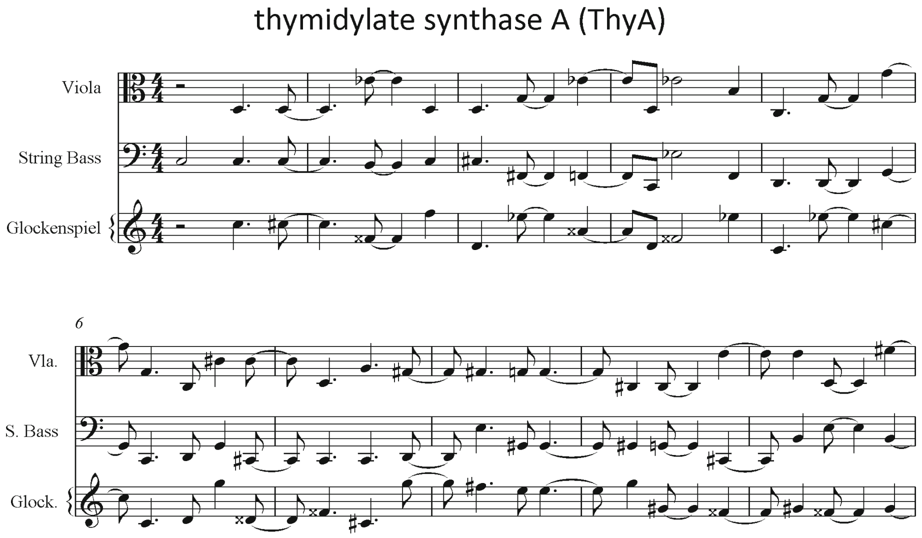


Fig. 6. Score of the musical translation of the chain A of the human Thymidylate Synthase A (ThyA). The three instruments represent respectively the primary sequence (Glockenspiel) and the two different amino acids correlated in the tertiary structure (String Bass, Viola).

4 Discussion

We obtained a polyphonic music by translating into musical notes the amino acid sequence of a peptide (the chain A of ThyA) and arranging them in chords by analyzing their spatial relationship (see Figure 7). To our knowledge, it is the first time that a team of scientists and musicians creates a polyphonic music that describes the entire 3D structure of a bio-molecule.

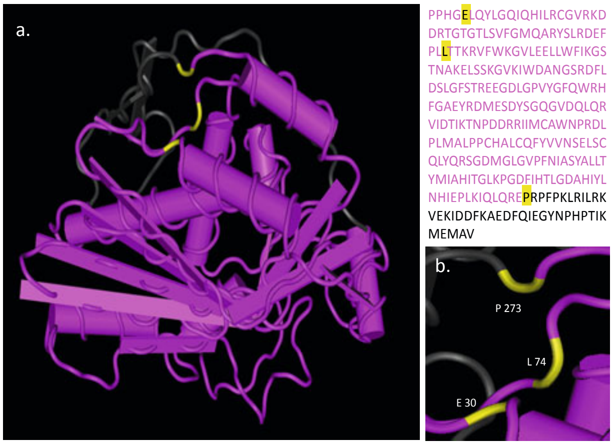


Fig. 7. From tertiary protein structure to musical chord. The primary structure of ThyA (chain A), on top right of the figure, fold into its tertiary structure (a, b). In yellow an example of the amino acids composing a musical chord: E30, L74 and P273 are non-obviously linked amino acids accordingly to our 3D spatial analysis criteria.

Previous works, attempting to translate a protein in music, focused on primary or secondary protein structure and used different tricks to obtain a polyphonic music.

Instead, the Matlab® script we developed is able to analyze the PDB file that contains the spatial coordinates of each atom composing the amino acids of the protein. The computation of distances and other useful geometrical properties between non-adjacent amino acids, generates a MIDI file that codifies the 3D structure of the protein into music.

In this way, the polyphonic music contains all the crucial information necessary to describe a protein, from its primary to its tertiary structure. Nevertheless, our analysis is fully reversible: by applying the same translation rules that are used to generate music, one can store, position by position, the notes (i.e. the amino acids) and obtain their distance. A first order musical sequence gives not enough information to recover the true protein structure, because there is more than one unique possibility to draw the protein. On the contrary, our approach, based on a third order musical sequence, has 3 times more data and describes one and only one solution to the problem of placing the amino acids in a 3D space.

5 Conclusions

Our work represents an attempt to communicate to a wider audience the complexity of the 3D protein structure, based on a rhythmic musical rendering. Biomusic can be an useful educational tool to depict the mechanisms that give rise to intracellular vital signals and determine cells fate. The possibility to "hear" the relations between amino acids and protein folding could definitely help students and a non technical auditory to understand the different facets and rules that regulate cells processes.

Moreover, several examples of interdisciplinary projects demonstrated that the use of an heuristic approach, sometimes perceived by the interacting audience as a game, can lead to interesting and useful scientific results [2, 5, 16]. We hope to bring musical harmony rules into the proteomic research field, encouraging a new generation of protein folding algorithms. Protein structure prediction, despite all the efforts and the development of several approaches, remains an extremely difficult and unresolved undertaking. We do not exclude that, in the future, musicality could be one of the driving indicators for protein folding investigation.

Acknowledgments. The authors would like to acknowledge Smart Collective (www. smart-collective.com) and Prof. Fabrizio Pirri (Politecnico di Torino) for supporting.

References

1. Ableton Live, http://www.ableton.com
2. Cyranoski, D.: Japan plays trump card to get kids into science. Nature 435, 726 (2005)
3. Dobson, C.M.: Protein folding and misfolding. Nature 426(6968), 884–890 (2003)
4. Dunn, J., Clak, M.A.: Life music: the sonification of proteins. Leonardo 32, 25–32 (1999)
5. Foldit, http://fold.it/portal
6. Gena, P., Strom, C.: Musical synthesis of DNA sequences. XI Colloquio di Informatica Musicale, 203–204 (1995)
7. Gena, P., Strom, C.: A physiological approach to DNA music. In: CADE 2001, pp. 129–134 (2001)
8. Gene2Music, http://www.mimg.ucla.edu/faculty/miller_jh/gene2music/home.html
9. GNMidi, http://www.gnmidi.com
10. Hayashi, K., Munakata, N.: Basically musical. Nature 310(5973), 96 (1984)
11. Hofstadter, D.R.: Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books, New York (1979) ISBN 0465026567
12. Jensen, E., Rusay, R.: Musical representations of the Fibonacci string and proteins using Mathematica. Mathematica J. 8, 55 (2001)
13. Mayor, S.: Serendipity and cell biology. Mol. Biol. Cell 21(22), 3808–3870 (2010)
14. Ohno, S., Ohno, M.: The all pervasive principle of repetitious recurrence governs not only coding sequence construction but also human endeavor in musical composition. Immunogenetics 24, 71–78 (1986)
15. Takahashi, R., Miller, J.: Conversion of amino acid sequence in proteins to classical music: search for auditory patterns. Genome Biology 8(5), 405 (2007)
16. The Space Game, http://www.thespacegame.org

Ludic Considerations of Tablet-Based Evo-Art

Simon Colton, Michael Cook, and Azalea Raad

Computational Creativity Group, Dept. of Computing,
Imperial College, London
http://www.doc.ic.ac.uk/ccg

Abstract. With the introduction of the iPad and similar devices, there is a unique opportunity to build tablet-based evolutionary art software for general consumption, and we describe here the i-ELVIRA iPad application for such purposes. To increase the ludic enjoyment users have with i-ELVIRA, we designed a GUI which gives the user a higher level of control and more efficient feedback than usual for desktop evo-art software. This relies on the efficient delivery of crossover and mutation images which bear an appropriate amount of resemblance to their parent(s). This requirement in turn led to technical difficulties which we resolved via the implementation and experimentation described here.

1 Introduction

While interacting with evolutionary art software can be a very rewarding experience, doing so is not yet a mainstream hobby, with the notable exception of online collaborations such as the Electric Sheep project [1]. The recent proliferation of tablet devices such as the iPad – where an increased emphasis is put on users enjoying their interaction with software – offers a good opportunity to bring evolutionary art to a wider audience. We have built the j-ELVIRA desktop evolutionary art program (which stands for (J)ava (E)volution of (L)udic (V)ariation in (R)esponsive (A)rtworks), as a rational reconstruction of the Avera software [2], in the usual mould of human-centric evo-art software such as NEvAr [3]. Porting j-ELVIRA to the iPad raised more than just issues related to re-programming the software. In particular, the touchscreen interface and the expectation of constant, enjoyable interaction with iPad applications required a new design for the GUI and improvements to the efficiency and quality of image generation. We provide details here of the iPad implementation (i-ELVIRA), including aspects of the interface design and the nature of image generation, in addition to some experiments we have performed to inform our design choices.

In section 2, we describe the particle-based image generation underpinning the ELVIRA systems, and we critique the desktop version from the perspective of user-interaction. This critique led us to design i-ELVIRA in such a way as to increase the *ludic quality* of the software, i.e., how much fun it is to interact with. These design choices led to certain technical difficulties. In particular, we found that images were generated too slowly to achieve the kind of ludic interaction we hoped for, and that often the images were not appropriate, i.e., they were

© Springer-Verlag Berlin Heidelberg 2011

blank, or bore too much/too little resemblance to their parents. In sections 3 and 4, we describe how we improved both the generation speed and quality of the images respectively, including details of some experiments we undertook to assess mutated image fidelity. In section 5, we assess the suitability of i-ELVIRA for mainstream use, and we discuss some future directions for our work.

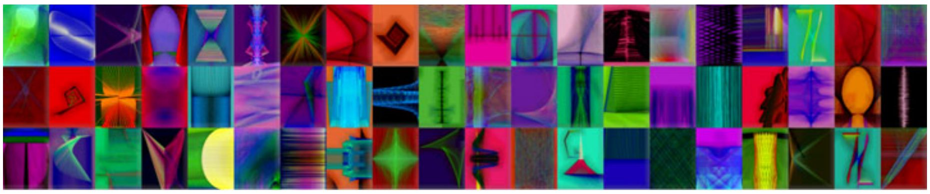
2 A Tablet-Based Interface Design

Particle based image generation schemes have been used to good effect in evolutionary art, for instance in the generation of ricochet compositions [4], and within the neuro-evolution framework described in [5]. In [2], Hull and Colton introduced an image evolution scheme which we have re-implemented into the j-ELVIRA software. The scheme uses six *initialisation trees* to control the nature of P particles, in terms of a sextuplet defining their location and their colour: $\langle x, y, r, g, b, a \rangle$. Over T timesteps, each particle changes colour and position, as controlled by six corresponding *update trees*, and a line is drawn from their previous position to their new one. The genomes of images therefore comprise a background colour which the canvas is filled with initially, and 12 trees which represent mathematical functions that calculate collectively the colour and position of a particle numbered p as it is initialised and as it changes at timestep t. Each update function takes as input the current $\langle x, y, r, g, b, a \rangle$ values of the particle in addition to t and p, all of which may be used in the function calculations. The canvas onto which lines are drawn is defined by the rectangle with corners $(-1, -1)$ and $(1, 1)$. At each timestep, after all the particle lines have been rendered, a Gaussian blur is applied. Each new set of lines is drawn on top of the previously resulting blurred background.

With the default values of $P = 1000$ and $T = 100$, the production of an image requires the plotting of 100,000 lines and 100 blurring operations. In [2], the authors barely explored the variety of images which can be produced, because their implementation was slow due to the drawing (off-canvas) of very long lines. By truncating – if necessary – the start and end points of the lines to be within the canvas rectangle, we achieved a much faster implementation. This allowed greater exploration of the types of images that can be produced, and we have seen a huge variety of images which can evoke perception of: lighting effects, shadowing, depth, texture and painterly effects. Moreover, the images produced often have a non-symmetrical and moderately hand-drawn look, which can add to their appeal. A sample of sixty images produced using this method is given in figure 1, along with an example genome (of 12 flattened trees) and the resulting image. We see that the complexity of the images derives not from the complexity of the individual trees, but rather from the iterative nature of the calculations the trees perform, and the blurring effect, which adds much subtlety to the images.

j-ELVIRA has a user interface similar to that of many evo-art packages:

- To start with, the user waits while a series of randomly generated genomes are used to produce a user-chosen number of images (usually 25).



Initialisation functions	Update functions
$x(p)_0 = -(0.75/sin(p))/-p$	$x(p)_t = sin(0.25 - p)$
$y(p)_0 = -(p * -0.5) + (p/-0.01)$	$y(p)_t = -sin((x_{t-1}(p) - 0.001) * (a_{t-1}(p)/r_{t-1}(p)))$
$r(p)_0 = cos(cos(sin(-0.001/p)))$	$r(p)_t = sin(x_{t-1}(p)) + 0.75$
$g(p)_0 = -sin(sin(0.001 + 100 * p))$	$g(p)_t = sin((-b_{t-1}(p) - y_{t-1}(p)) * t)$
$b(p)_0 = (-p * p)/(-0.25 * p)$	$b(p)_t = y_{t-1}(p) * -0.5$
$a(p)_0 = sin(-0.01)$	$a(p)_t = cos(t) * 2 * r_{t-1}(p)$

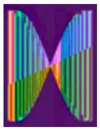


Fig. 1. Top: 60 exemplar images produced with the particles method. Below: a genotype/phenotype pair in terms of the genome functions and the image produced.

• The user selects the images they prefer, or sometimes at the start of the session, they tend to select any images which are in any way unusual, as many of the images will be devoid of interest.
• The user chooses to either crossover and/or mutate the selected images. They then wait while the software chooses randomly from the selected individuals and crosses pairs of them and/or mutates them into child genomes from which new images are produced and shown to the user.

When implementing i-ELVIRA in Objective C for the iPad, we referred to Apple's iPad development guidelines, where it is noted that: (a) "People, not apps, should initiate and control actions ... it's usually a mistake for the app to take decision-making away from the user" (b) "People expect immediate feedback when they operate a control", and (c) "When appropriate, add a realistic, physical dimension to your app". We felt that the interaction design for j-Elvira was at odds with guidelines (a) and (b). In particular, by choosing which individuals from the preferred set to crossover/mutate, j-Elvira takes too much control away from the user, thus contradicting guideline (a). Moreover, j-Elvira forces users to wait much too long (measured in minutes) for the production of 25 images before more progress can be made, thus contradicting guideline (b). To address these issues, we designed the iPad interface so that the user explicitly chooses which individual to mutate, and which pairs to crossover. As soon as they have made their choice, the child images are produced immediately. This hands back more control to the user, and they are supplied with feedback as soon as a new image can be generated (measured in seconds). In response to guideline (c), to add a realistic element to the interface, we used metaphors of: a recycling tray for discarding images; a printer/scanner for copying and generating images; and rows of trays into which sets of images can be dragged.

The random generation of images at the start of a session with j-ELVIRA is also unappealing, as it is prone to producing blank/dull images, as all/most of the particle lines are drawn off-canvas. To address this, we generated tens of thousands of images randomly, and chose by hand 1,000 *preset images* from them

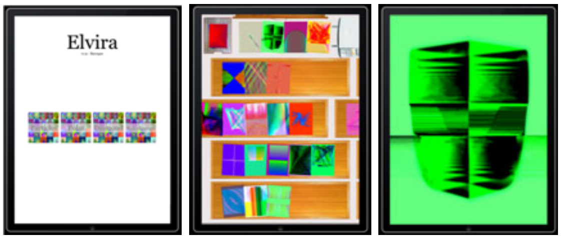


Fig. 2. (i) Opening screen, where old sessions can be loaded (ii) browsing screen, where images are generated and organised into trays (iii) editing screen, where images can be scaled, rotated, translated, cropped and re-rendered

(60 of which are given in figure 1). These were chosen to maximise the variety of images afforded by the particles method, so that hopefully every user may find a preset which fits their aesthetic preferences. We have performed extensive testing, and we have found that neither large trees, nor more complex functions than cosine and sine, nor more complex programmatic structures such as conditionals lead to more interesting or varied images. Therefore, when generating the preset images, we restricted the random search to trees of size 12 or less, containing only the arithmetic and trigonometric functions and the terminals $\{0, 0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 10, 100\}$ and their negations.

We provide screenshots of the graphical user interface for i-ELVIRA in figure 2. In overview, the user is presented with a continuous supply of preset images at the top of the screen. The user can drag these images into trays lower on the screen to organise them. The rows of trays add physicality to the design and enable the choosing of any pair of images for crossover. If a user drags one image on top of another, i-ELVIRA immediately crosses over the genomes of the two images to produce four child images, which appear at the top of the screen. Moreover, if the user drags an image to the printer/scanner, i-ELVIRA will immediately start the generation of four mutations of the image. The images are produced at the full resolution of the iPad screen (768×1024 pixels), and by tapping on an image, it expands to fill the whole screen. In this state, the image can be cropped, scaled, rotated and translated, and these transformations are recorded in the image's genome (as a 4×4 matrix). The user can choose to re-render the image to further explore it, which is done by applying the transformation matrix to the position of the particle just before the line between its old (transformed) position, and its new one is rendered.

The interactive nature of the GUI for i-ELVIRA forced two issues. Firstly, users did not expect to wait long for image generation, i.e., they wanted near-immediate gratification in the form of more images in the style of ones they have chosen. Secondly, people expected the images generated in response to their choices to always be *appropriate* to those choices, i.e., that the children of crossed-over or mutated individuals should resemble their parents, but not too much. Unfortunately, our first implementation of i-ELVIRA regularly presented the user with inappropriate images, and took up to 15 seconds to produce each one. In early experiments, we found that this uncertainty in child fidelity and the slow execution times largely ruined the ludic appeal of the software, and hence both issues had to be addressed, as described in the following sections.

3 Efficient Image Rendering

The printer/scanner metaphor helps disguise the time taken to produce images, because as soon as a user has chosen an image for mutation or a pair of images for crossover, an animation of a blank sheet of paper being output holds the user's attention for a few seconds. However, we estimated that any longer than a five second wait for an image to be produced would be detrimental the the user's enjoyment. To calculate the lines which comprise an image, 6 functions have to be called 100,000 times to calculate the position and colour of the particles. Fortunately, our preset genomes contain fairly succinct functions, due to the fact that we restricted tree size to 12 nodes or fewer. However, we still found that the calculations took prohibitively long: around eight seconds on average. This was because we were representing the functions as trees which were being interpreted at runtime. We chose to flatten the trees into mathematical functions such as those in figure 1 and precompile these into i-Elvira. This dramatically reduced the calculation time for the particles to around half a second on average. Of course, a drawback to precompilation is a reduction in the size of the search space, as new trees cannot be generated at runtime, nor existing ones altered. In particular, the only option for crossover is to swap entire initialisation/update functions of two parents, and for mutation, it is to randomly substitute one or more function with ones from other individuals (i.e., no random generation of trees is possible). However, the trees themselves are fairly small, so there wasn't much scope for crossing over subtrees anyway. Moreover, from the 1000 preset images, we extracted 1798 distinct initialisation functions and 2076 distinct update functions. Hence, given that any initialisation function may be swapped for any other, and likewise for update functions, $1798^6 \times 2076^6 = 2.7 \times 10^{39}$ distinct genomes can be produced, which is more than enough.

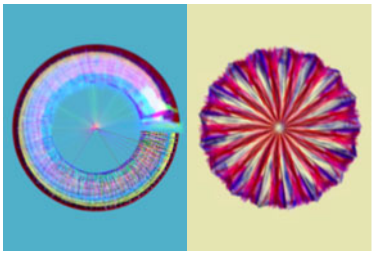
Having halved the image generation time through precompilation, we turned to the other major bottleneck: the rendering of the image, i.e., the drawing of the lines and the blurring operation. Apple supplies the 2D iPad *CoreGraphics* graphics library. In our first attempt, we employed CoreGraphics to draw the lines and wrote a per-pixel blurring operation, which changes a pixel's colour to be an average of those in a neighbourhood around it – with a bigger neighbourhood producing a more blurred image. Sadly, this method was too inefficient for our purposes, as it took around 6 seconds to render the image. Hence, we made several improvements to the image generation pipeline in order to optimise the process. The most important of these was using OpenGL ES 1.1, an early mobile version of the popular graphics engine, instead of CoreGraphics. To make the move to OpenGL, we altered the rendering process to employ a vertex-based drawing model, whereby each rendering pass contains a single update to the particles which are described in terms of start and end vertices.

Recall that at each timestep, a blur operation is performed in the image generation process. As OpenGL ES 1.1 does not support pixel shaders, which would have allowed for a per-pixel Gaussian blur to be applied between passes, we instead pass the base image (representing a composite of each rendering stage completed so far) to OpenGL as a texture. After the lines for the current timestep are drawn on top of this texture, a further composite comprising the base image and the new lines is drawn out to another texture using an OpenGL FrameBufferObject. To this second texture, we perform a blur by redrawing the texture object four times, offset by one pixel in the four compass directions, at a reduced opacity. This produces a blurred image without being too costly for OpenGL to draw. The resulting five-layer image is then flattened down to become the base image for the next rendering pass. This new pipeline reduced the rendering time to around 3.5 seconds on average, which is better than the 15 seconds we started with, and within our 5 second ludic limit.

4 Generation of Appropriate Images

The second issue raised by the interactive nature of i-ELVIRA was the disappointment users felt when they were presented with an image which was either blank, or looked different to what they expected (i.e., too similar or dissimilar to its parent). Recall that four mutations of a chosen image are supplied when the user makes a choice, and similarly four offspring are supplied when the user chooses two parents to crossover. Noting that efficiency of image generation is a major issue, we decided not to perform a post-hoc check on image quality, in order to reject an image on grounds of low quality, as this would mean producing another one, and therefore at least doubling the image generation time on occasions. Instead, we concentrated on enabling i-ELVIRA to more reliably generate genomes that would produce appropriate images. Blank or nearly blank images are caused by a lack of lines being drawn on the canvas. One way to avoid the generation of such images altogether is to somehow map the position of each particle at each timestep to somewhere within the canvas. One possibility is to map x and y to their fractional parts, whilst maintaining their parity. Unfortunately, this produces largely uninteresting images, as each line is rendered on the canvas, and many images gain their beauty by having fewer than the total 100,000 lines drawn. For instance, many interesting images exhibit a blurry look, as no lines are drawn on them for the last 10 or 20 timesteps.

However, we did find a number of mappings that regularly produce pleasing images. Two such mappings are given along with sample images produced using them in figure 3. Note that $f(k)$ denotes the fractional part of k, and $\theta_{p,s} = \frac{2\pi(p \bmod s)}{s}$ for a parameter s, which determines the number of segments in the kaleidoscope image (default 17). Given that the images produced by these mappings have appeal, and that the extra processing does not produce a noticeable increase in rendering time, we have enabled i-ELVIRA to generate images using the mappings, and a number denoting which mapping to use is stored in the genome of every generated image. We accordingly added new genomes which use these mappings, to i-ELVIRA's presets. In addition, we looked at the unmapped genotypes which produced the blank images, and we realised that most



Polar mapping: $\begin{pmatrix} x_t(p) \\ y_t(p) \end{pmatrix} \mapsto \begin{pmatrix} f(x_t(p))cos(2\pi f(y_t(p))) \\ f(x_t(p))sin(2\pi f(y_t(p))) \end{pmatrix}$

Kaleidoscope mapping: $\begin{pmatrix} x_t(p) \\ y_t(p) \end{pmatrix} \mapsto$
$\begin{pmatrix} cos(\theta_{p,s}) & sin(\theta_{p,s}) \\ -sin(\theta_{p,s}) & cos(\theta_{p,s}) \end{pmatrix} \begin{pmatrix} f(x_t(p))cos(\frac{2\pi}{s} f(y_t(p))) \\ f(x_t(p))sin(\frac{2\pi}{s} f(y_t(p))) \end{pmatrix}$

Fig. 3. Example images produce by the polar and kaleidoscope particle mappings

of them were caused by the update functions for the x and/or y co-ordinates being constant. Hence, we gave i-ELVIRA the ability to avoid producing child genomes through mutation or crossover where either the x or y update function was not dependent on any input value. We found that this drastically reduced the number of blank or nearly blank images to an acceptable level.

People find it difficult to predict what the children of two parent images will look like, and are fairly forgiving when crossover images don't resemble their parents. Indeed, people tend to use the crossover mechanism to search the space of images, rather than to focus on a particular style. We experimented with different crossover mechanisms, until the following emerged as a reliable way to produce child images: given two parent images A and B, child C inherits the background colour of B, and five initialisation functions and four update functions from A, with the missing initialisation and update functions inherited from B. This mechanism works well, except when people crossover two images which are themselves the children of a shared parent. In this case, there was a tendency for C to be very similar to A and/or B. Hence, whenever an offspring genome is produced, if the parents share 10 or more functions, the offspring is mutated by swapping one initialisation function for a randomly chosen one, and similarly swapping two update functions. This produces children which vary enough from their parents. In producing four offspring, i-ELVIRA produces two children as above, and two children with the contributions of A and B swapped.

Users of i-ELVIRA tend to be much less forgiving for mutated versions of their chosen images, as mutation is the major way of exerting fine-grained control over image production. Hence users tend to be quite disappointed when a mutated image is too dissimilar or too similar to its parent. Due to the precompiled nature of the functions in i-ELVIRA, the smallest mutation possible is a swapping of a single initialisation function for a randomly chosen one. However, we found that mutating only initialisation functions was prone to producing too many *twin images*, i.e., pairs of siblings in the four produced by i-ELVIRA that look too similar (see the experimental results below). We therefore looked at mutating a single update function, but we found that this could sometimes produce both too dissimilar mutations and too similar ones. Fortunately, we found that both these problems could be managed by enabling i-ELVIRA to perform a *dependency analysis* on the functions in the genomes of images. Looking at table 1, which

Table 1. Dependency analysis for the genome from figure 1

	x	y	r	g	b	a		
x	1	0	0	0	0	0	1	$x(p)_t = sin(0.25 - p)$
y	1	1	1	0	0	1	4	$y(p)_t = -sin((x_{t-1}(p) - 0.001) * (a_{t-1}(p)/r_{t-1}(p)))$
r	1	0	1	0	0	0	2	$r(p)_t = sin(x_{t-1}(p)) + 0.75$
g	1	1	1	1	1	1	6	$g(p)_t = sin((-b_{t-1}(p) - y_{t-1}(p)) * t)$
b	1	1	1	0	1	1	5	$b(p)_t = y_{t-1}(p) * -0.5$
a	1	0	1	0	0	1	3	$a(p)_t = cos(t) * 2 * r_{t-1}(p)$
	6	3	5	1	2	3		

re-iterates the update functions for the genome in figure 1, we see, for example, that the updated alpha value $a(p)_t$ for particle number p at timestep t is dependent on the previous red value of p, namely $r_{t-1}(p)$. However, looking at $r_t(p)$, we see that the updated red value of p is dependent on the previous x co-ordinate of p, namely $x_{t-1}(p)$. Working backwards, we can therefore conclude that output of $a(p)_t$ is dependent on output of the r and x update functions.

For each precompiled update function, we gave i-ELVIRA information about which other functions appear locally, e.g., it is told that $a(p)_t = cos(t)*2*r_{t-1}(p)$ is locally dependent on $r_{t-1}(p)$. We further implemented a routine to work backwards from the local dependencies to determine which variables each function is ultimately dependent on. Hence, in the example in table 1, i-ELVIRA knows that mutating the $r_t(p)$ or $x_t(p)$ update function will also affect the output from the $a_t(p)$ function. In effect, i-ELVIRA can build a dependency matrix such as that in table 1, where a 1 in row w and column c indicates a dependency of the row w function on the column c function. For instance, the 1 in the g row and b column indicates that the update function $g(p)_t$ is dependent on previous b values. Note that we put a 1 in each diagonal position, because each function is dependent on itself (in the sense that mutating a function will naturally alter that function). The row totals indicate how many update functions that row's function is dependent on. The column totals indicate how many update functions depend upon that column's function, and can therefore be used to indicate how disruptive changing that function will be to the location and colour of the particles. We call these numbers the *dependency quotients* for the functions. For instance, the 5 in the r column of table 1 indicates that mutating $r(p)_t$ will effect 5 attributes of each particle, namely their y, r, g, b and a values.

On inspection of a number of mutated individuals where a single update function was swapped for a randomly chosen one, we found that mutating an update function which had a dependency quotient of 1 led to images which were too similar to the original. In contrast, swapping a function which had a dependency quotient of 6 led to a change in every colour and location aspect of each particle, and hence led to a fundamentally different image, i.e., too dissimilar to the original. So, for instance, for the genome in table 1, it would be unwise to mutate either the $x_t(p)$ update function, upon which all the other functions depend, or the $g_t(p)$ update function, upon which no other functions depend. We also analysed a number of other cases where the mutation produced was inappropriate, and used our findings to derive a heuristic mutation mechanism which produces an acceptably high proportion of appropriate mutations.

This mechanism swaps a single update function for a randomly chosen one. It first tries to swap the r, g, b or a function, but will only chose one if it has a dependency quotient between 2 and 5 inclusive. If this fails, it attempts to swap the x or y function, but again only if one has a dependency quotient between 2 and 5 inclusive. If this fails, then an update function is chosen randomly and swapped, ensuring that neither the $x_t(p)$ nor the $y_t(p)$ update function is swapped for a constant function. Moreover, if neither the x nor y function is dependent on the r, g, b or a functions, then either the x or the y initialisation function (chosen randomly) is mutated. Each part the heuristic mechanism was motivated by analysis of the reasons why a set of mutations produced inappropriate images. To determine the value of the heuristic mechanism, we compared it to swapping four, five and six initialisation functions for randomly chosen ones (called the *4-init*, *5-init* and *6-init* methods respectively), and swapping a single update function for a randomly chosen one, with no dependency analysis or constant function checking (called the *1-update* method).

Table 2. Results for 250 sets of 4 images by 5 mutation methods

Method	n_b	n_s	n_d	n_t	p_f
Heuristic	6	30	25	29	0.84
1-update	12	34	95	19	0.63
4-init	8	80	68	128	0.66
5-init	12	47	99	114	0.69
6-init	17	33	114	128	0.69

We chose 250 of i-ELVIRA's presets randomly, and for each method, we took each preset in turn and produced 4 mutations for it, as this is the number that users of i-ELVIRA are presented with. We performed image analysis on the resulting 1000 mutated images to determine how appropriate they were. We first recorded the number of mutated images which were essentially blank (n_b). We found that two images look similar if they have a similar colour distribution, and/or if they exhibit a similar shape, i.e., the parts of the canvas which are covered by lines for the two images are similar. Given two images i_1 and i_2, we implemented a method to determine the colour distance, $d_c(i_1, i_2)$ of the images in terms if their colour histograms, and a method to determine the shape distance $d_s(i_1, i_2)$, in terms of the positions in a 24×24 grid which are crossed by particle lines. Both methods return a value between 0 and 1, with 0 indicating equal images, and 1 indicating as different as possible images. Analysing pairs of images visually, we settled on two definitions: a pair of images i_1 and i_2 are *too similar* if $min(d_c(i_1, i_2), d_s(i_1, i_2)) < 0.1$, and *too dissimilar* if $max(d_c(i_1, i_2), d_s(i_1, i_2)) > 0.9$.

Within the 1000 images produced for each mutation method, we recorded the number of mutations which were too similar to their parent (n_s) and the number which were too dissimilar (n_d). We also recorded the number of twins produced (n_t). Finally, we recorded the proportion, p_f, of sets of four mutations which contained no inappropriate images (i.e., neither too similar to the parent or eachother, nor too dissimilar to the parent). The results for the five methods are given in table 2. We see that the heuristic method outperforms the other methods in all the metrics, with the exception of the 1-update method producing fewer twins (at the cost of an increased distance between child and parent). Also, as previously mentioned, we see that the mutation methods which alter only initialisation functions all suffer from producing too many twins.

5 Conclusions and Future Work

We have developed the i-ELVIRA evolutionary art application for the iPad, guided by general ludic considerations, which include: enabling constant user-interaction with no periods where the user is waiting for the software to finish processing; avoiding supplying the user with uninteresting or inappropriate images; a natural interaction design which enables the crossing over and mutation of chosen images; an ability for users to customise their artworks and for them to share their creations with others. We are currently finalising i-ELVIRA for distribution, which requires a ludic graphical user interface to the evolution and image generation mechanisms. This has only been made possible because we reduced the image rendering time to 3.5 seconds, and we increased the reliability with which appropriate mutation images are produced. Looking at the results in table 2, we see that the heuristic mutation method delivers a set of four appropriate mutations with an 84% likelihood, which we believe is acceptable for i-ELVIRA. In the context of evolving buildings for a video game, producing artefacts which have an appropriate resemblance to their parents was addressed in [6]. This is a key question in bringing evolutionary art to the general public.

To hopefully improve i-ELVIRA, we will experiment with showing users updates during the rendering process, which might hold their attention (although we have found that for many images, this process can be quite dull, as all the lines are drawn at the start or the end of the process). We will also experiment with different blurring processes to explore different visual styles, and we will enable a search mechanism so that people can find presets similar to ones they like. With all our experimentation, we will undertake extensive user testing to determine the value of the changes we impose. In particular, using i-ELVIRA and j-ELVIRA as research tools, our next step will be to conduct user studies, whereby we try and derive methods for estimating people's reactions to images. Ultimately, we aim to embed machine learning methods into evolutionary art software, so that it can approximate people's aesthetic considerations and use this to deliver better images, as we began to investigate in [7] for image filtering. In the long term, we aim to show that sophisticated user modelling techniques can lead to more enjoyable software such as i-ELVIRA for public consumption, and also be a driving force for machine learning.

Acknowledgements

We would like to thank the anonymous reviewers for their interesting comments. We would also like to thank the Department of Computing at Imperial College, London for funding the implementation of the i-ELVIRA system.

References

1. Draves, S.: The electric sheep screen-saver: A case study in aesthetic evolution. In: Proceedings of the EvoMusArt Workshop (2005)
2. 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)
3. 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)
4. Greenfield, G.: Evolved ricochet compositions. In: Proceedings of the EvoMusArt Workshop (2009)
5. Hastings, E., Guha, R., Stanley, K.: Neat particles: Design, representation, and animation of particle system effects. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2007)
6. Martin, A., Lim, A., Colton, S., Browne, C.: Evolving 3D buildings for the prototype video game subversion. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010. LNCS, vol. 6024, pp. 111–120. Springer, Heidelberg (2010)
7. Colton, S., Torres, P., Gow, J., Cairns, P.: Experiments in objet trouvé browsing. In: Proc. of the 1st International Conference on Computational Creativity (2010)

Evolving Art Using Multiple Aesthetic Measures

E. den Heijer[1,2] and A.E. Eiben[2]

[1] Objectivation B.V., Amsterdam,
The Netherlands
[2] Vrije Universiteit Amsterdam,
The Netherlands
eelco@objectivation.nl, gusz@cs.vu.nl
http://www.cs.vu.nl/~gusz/

Abstract. In this paper we investigate the applicability of Multi-Objective Optimization (MOO) in Evolutionary Art. We evolve images using an unsupervised evolutionary algorithm and we use two aesthetic measures as fitness functions concurrently. We use three different pairs from a set of three aesthetic measures and we compare the output of each pair to the output of other pairs, and to the output of experiments with a single aesthetic measure (non-MOO). We investigate 1) whether properties of aesthetic measures can be combined using MOO and 2) whether the use of MOO in evolutionary art results in different images, or perhaps "better" images. All images in this paper can be viewed in colour at http://www.few.vu.nl/~eelco/

1 Introduction

One of the fundamental problems in the field of evolutionary art is the issue of fitness assignment. Within evolutionary art there are two possible ways to assign fitness to an artefact; the first option is to delegate fitness assignment to a human being in an interactive evolutionary setup (Interactive Evolutionary Computation or IEC). Setting up an IEC environment to evaluate art, music or other artefacts is relatively simple, and IEC has been applied successfully in a wide variety of application domains (especially in domains where computational fitness functions are hard to come by) such as art, graphic design, music and many others [16]. IEC also has a number of drawbacks; the most important one is user fatigue, whereby the user that steers the evolution process (by evaluating artefacts) becomes tired and/ or loses interest ("fatigued"). This implies that typical IEC experiments have relatively small populations and relatively few iterations and this severely limits the potential output of any IEC setup. The other way of fitness assignment within evolutionary art is unsupervised evolutionary art, whereby a computational fitness function assigns a score to an artefact without human intervention. The creation of fitness functions for the evaluation of art is regarded as one of the open problems in evolutionary art [9]. In previous work we investigated the use of six aesthetic measures as fitness functions [6] [7]. We showed that the choice of the aesthetic measure has a significant impact on the style of the evolved images.

© Springer-Verlag Berlin Heidelberg 2011

1.1 Research Questions

In this paper we investigate whether it is possible to combine the effect of multiple aesthetic measures concurrently using a Multi-Optimization Evolutionary Algorithm (MOEA). In previous work we have shown that the choice of the aesthetic measure significantly determines the "style" of the generated art [6] [7]. With MOEA, we want to investigate whether the influence of different aesthetic measures can be combined into the same image. For example, if we use one aesthetic measure that focuses on the use of contrast in an image, and one aesthetic measure that focuses on certain color transitions within an image, then we would like to evolve images that have both properties. So our first research question is; can we combine the effects from multiple aesthetic measures into the same image using a MOEA? Second, we want to know whether the use of a MOEA results in "better" images in evolutionary art. Beautiful images often have multiple "good" properties; good use of contrast, interesting color transitions, good level of interestingness (not too simple, not too complex/ chaotic) etc. If we evolve images by optimizing multiple objectives simultaneously, it should – in theory – lead to "better" images.

The rest of the paper is structured as follows. First we discuss evolutionary art and the use of MOEA in evolutionary art (section 2). Section 3 briefly discusses our software environment Arabitat. Next, we describe the experiments and their results in section 4. Section 5 contains conclusions and directions for future work.

2 Evolutionary Art

Evolutionary art is a research field where methods from Evolutionary Computation are used to create works of art (good overviews of the field are [11] and [2]). Some evolutionary art systems use IEC or supervised fitness assignment (e.g. [15], [12]), and in recent years there has been increased activity in investigating unsupervised fitness assignment (e.g. [5], [13]).

2.1 The Use of MOEA in Evolutionary Art

MOEA's have not been used frequently in the field of evolutionary art; in [14] Ross & Zhu describe research into evolving procedural art by comparing evolved images with a target image. The fitness functions in their MOEA setup are distance metrics that calculate the difference between an individual and the target image. Our approach is different since we do not evolve images with a target image in mind. Our approach is more similar to [5] in which Greenfield evolves images and fitness components concurrently in a co-evolution setup. Our approach is different in two ways; first, we do not use co-evolution in our experiments, and second, we have a number of "fixed" aesthetic measures that we use as the fitness functions.

Table 1. Evolutionary parameters of our evolutionary art system used in our experiments

Symbolic parameters	
Representation	Expression trees, see table 2
Initialization	Ramped half-and-half (depth between 2 and 5)
Survivor selection	Tournament, Elitist (best 3)
Parent Selection	Tournament
Mutation	Point mutation
Recombination	Subtree crossover
Fitness function	Multiple aesthetic measures (see 2.2)
Numeric parameters	
Population size	200
Generations	20
Tournament size	3
Crossover rate	0.9
Mutation rate	0.1
Maximum tree depth	8

2.2 Aesthetic Measures

The aesthetic measures that we use in this paper have different mechanisms and backgrounds, and we will describe them briefly. For a more detailed description we refer to the original papers. We will briefly describe the aesthetic measures Benford Law, Global Contrast Factor, and Ross & Ralph Bell Curve.

Benford Law. We use an aesthetic measure based on Benford Law [1]; Benford Law (or first-digit law) states that list of numbers obtained from real life (i.e. not created by man) are distributed in a specific, non-uniform way. The leading digit occurs one third of the time, the second digit occurs 17.6%, etc. We use the Benford Law over the distribution of brightness of the pixels of an image. We used the same implementation and settings as in previous experiments so we refer to [7] for details.

Global Contrast Factor. The Global Contrast Factor is an aesthetic measure described in [8]. Basically, the global contrast factor computes contrast (difference in luminance or brightness) at various resolutions. Images that have little or few differences in luminance have low contrast and are considered 'boring', and thus have a low aesthetic value. We used the same implementation and settings as in previous experiments so we refer to [7] for details.

Ross and Ralph (bell curve). A second aesthetic measure that we implemented is Ross & Ralph [13]. This measure is based on the observation that many fine art painting exhibit functions over colour gradients that conform to a normal or bell curve distribution. The authors suggest that works of art should have a reasonable amount of changes in colour, but that the changes in colour should reflect a normal distribution (hence the name 'Bell Curve'). The computation takes several steps and we refer to [13] for details.

3 Arabitat: The Art Habitat

Arabitat (Art Habitat) is our software environment in which we investigate evolutionary art. It uses genetic programming with Lisp expressions and supports both supervised and unsupervised evaluation. The details of Arabitat have been described in detail in [7] so we will not repeat it here. In addition to our system described in [7] we have implemented the Multi-Objective Optimization algorithms NSGA-II [3] and SPEA2. In this paper we will only discuss the experiments we did with NSGA-II. NSGA-II finds an optimal Pareto front by using the concept of non-domination; a solution A is non-dominated when there is no other solution that scores higher on all of the objective functions. Furthermore, NSGA-II uses elitism and a mechanism to preserve diverse solution by using a crowding distance operator. For more details, we refer to [3].

Function set. Many functions used are similar to the ones used in [15], [12] and [13]. Table 2 summarizes the used functions (including their required number of arguments);

Table 2. Function and terminal set of our evolutionary art system

Terminals	x,y, ephem_double, golden_ratio, pi
Basic Math	plus/2, minus/2, multiply/2, div/2, mod/2
Other Math	log/1, sinh/1, cosh/1, tanh/1, atan2/2, hypot/2, log10/1, squareroot/1, cone2/2, cone3/2, cone4/2
Relational	minimum/2, maximum/2, ifthenelse/3
Bitwise	and/2, or/2, xor/2
Noise	perlinnoise/2, fbm/2, scnoise/2, vlnoise/2, marble/2, turbulence/2
Boolean	lessthan/4, greaterthan/4
Other	parabol/2

The function set has already been described in detail in [7] so we will not repeat it here. There are a few new functions since [7] which we will describe briefly; *cone*2, *cone*3 and *cone*4 all draw circle-like patterns with the center in the middle (so the image looks like a cone from the top) and are variations on the *cone* function from [4].

4 Experiments

We did a number of experiments to evaluate the use of a MOEA in evolutionary art. First, we performed 10 runs with a single aesthetic measure (non-MOEA).

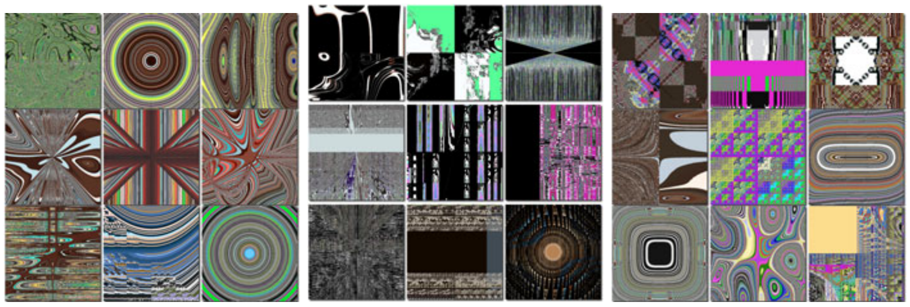


Fig. 1. Portfolio of images gathered from ten runs with Benford Law (left) and Global Contrast Factor (middle) and Ross & Ralph (right)

This resulted in 3 experiments (one for each aesthetic measure described in 2.2 consisting of 10 runs each. We hand-picked a number of images from these runs and created portfolios for each aesthetic measure. Basically, this is the same setup as the research we did described in [6] and [7], but we repeated these experiments since we altered the function set. Figure 1 shows three portfolios of the three experiments with a single aesthetic measure. As in [6] and [7] we see a clear difference in "style" between the three aesthetic measures. We will use Figure 1 (as a kind of benchmark) to evaluate the output of the experiments with the MOEA. Next, we performed three experiments with the NSGA-II algorithm [3] using 1) Benford Law and Ross & Ralph, 2) Global Contrast Factor and Ross & Ralph and 3) Benford Law and Global Contrast Factor. We did 10 runs with each setup, using the exact same experimental setup (evolutionary parameters from Table 1 and the function set from Table 2) except for the combination of aesthetic measures. From each run, we saved the Pareto front (the first front, with rank 0) and calculated the normalized fitness for image I for each objective f using $f_{normalized}(I) = f(I)/f_{average}$. This way, we normalized all scores between 0 and 1. Next, we ordered each individual on the sum of the normalized scores of the two objectives, and we stored the top 3 individuals from each run. With 10 runs per experiments, we have 30 individuals per experiment that can be considered the "top 30". Using this approach, we have a fair and unbiased selection procedure (since we did not handpick images for these selections). In the top 30 portfolio of the experiment with Benford Law and Ross & Ralph (Figure 2) we can clearly see the influence of both aesthetic measures in the images. The Benford Law aesthetic measures produces images with an organic, natural feel and the Ross & Ralph measure tends to produce image with a "painterly" feel (since it focuses on smooth transitions in colours). We can see these properties in most images and in some images they are combined (i.e. in the first three images in Figure 2). The last two images of the second row and the first image of the third row also appear in the close-up of the Pareto front in Figure 5. In the top 30 portfolio of the experiment with Global Contrast Factor and Ross & Ralph (Figure 3) we see again that the properties of both aesthetic measures appear in the images. GCF tends to produce images with stark contrast at various

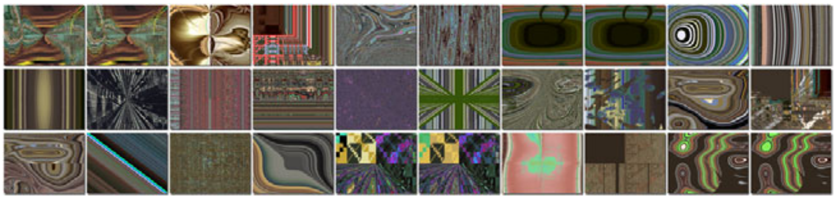


Fig. 2. Portfolio of images gathered from ten runs with NSGA-II with Benford Law and Ross & Ralph

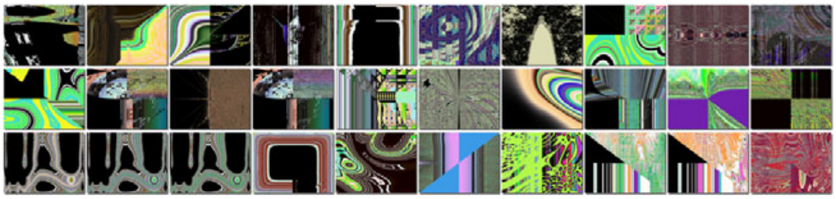


Fig. 3. Portfolio of images gathered from ten runs with NSGA-II with Global Contrast Factor and Ross & Ralph

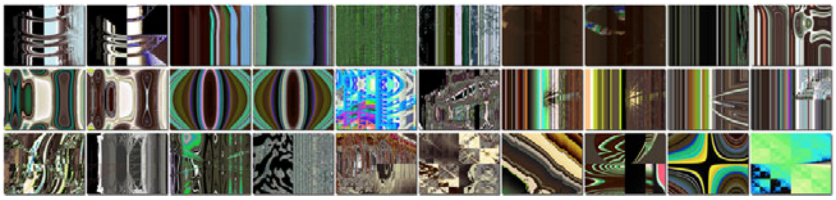


Fig. 4. Portfolio of images gathered from ten runs with NSGA-II with Benford Law and Global Contrast Factor

resolutions and Ross & Ralph tends to produce "painterly" images. If we compare this portfolio with the previous portfolio, we can clearly see more dark colours (especially black) in the images. This can be attributed to the influence of the GCF measure. There seems to be less "synergy" between the two measures; images either have a strong GCF signature or a strong Ross & Ralph signature, but few images have both. Apparently, it is difficult to mix the "styles" of these two aesthetic measures into one image. The 5th, 6th and 7th image of the second row appear in the close-up of the Pareto front in Figure 6. In the top 30 portfolio of the experiment with Benford Law with GCF (Figure 4) we clearly see the influence of the Global Contrast Factor; many images have a stark contrast and have dark areas. Nevertheless, if we compare these images with the portfolio of the Global Contrast Factor (Figure 1) then we can also detect the influence of the Benford Law aesthetic measure (although clearly not in all images). The Benford Law aesthetic measure produces images with a grainy, natural feel (see more images in [7]) and in a way these two properties seem to blend in a number of images (although not in all). It appears that these two aesthetic measures do not combine very well. The 2nd, 3rd and 4th image of the third row also appear in the close-up of the Pareto front in Figure 7.

4.1 Close-Ups of Pareto Fronts

We wanted to know in detail how a single Pareto front was organized, and whether we could see a gradual transition of the influence of measure A to measure B while moving over the Pareto front. We zoomed in on a single Pareto front and reconstructed the images that belong with each individual in that front. In the following figure we show the Pareto front for each pair of aesthetic measure (note that we did 10 runs per experiments, but we only show the Pareto front of one run). In Figure 5 we see the 15 individuals plotted based on their scores on the Ross & Ralph measures and the Benford Law measure. We normalized the scores between 0 and 1.

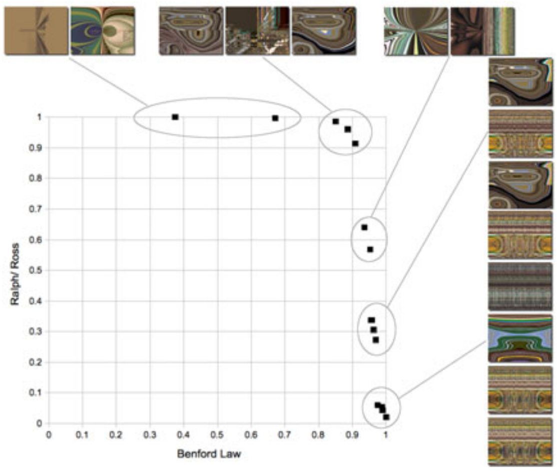


Fig. 5. Details of the Pareto front of Benford Law and Ross & Ralph with the corresponding images per element of the front

If we look at the individuals of the Pareto front in Figure 5, we can see a transition of the influence from aesthetic measure to the other. At the top we see “typical” Ross & Ralph images (we saw this type of images in Figure 1, right picture, and in [6]), and at the bottom/ right we see more typical Benford Law images. In between, at the right/ top we see the images where the influences blend most. Not that the images of the individuals in the upper right of the front (where the combined score is highest) are gathered in the Top 30 selection of Figure 2 (fourth row, first three images).

In Figure 6 we see the 12 individuals of a run of Ralph & Ross and the Global Contrast Factor. On the far left we see one individual that scores low on GCF and high on the Ralph & Ross measure. This image is a ‘typical’ Ralph & Ross

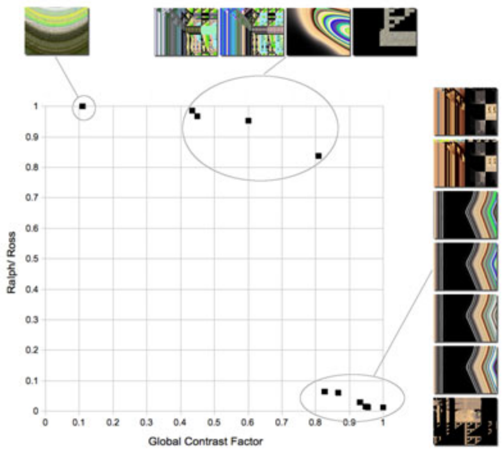


Fig. 6. Details of the Pareto front of Ralph & Ross and Global Contrast Factor with the corresponding images per element of the front

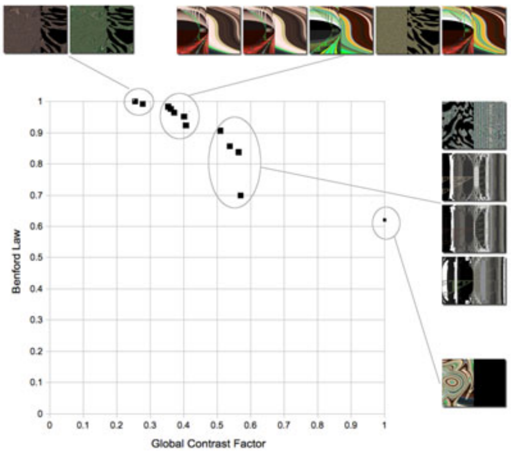


Fig. 7. Details of the Pareto front of Benford Law and Global Contrast Factor with the corresponding images per element of the front

image (we see similar images in Figure 1, right), and it is quite different from the cluster of images on the lower right; in this cluster we can clearly see the influence of the GCF measure, with heavy contrast and a lot of black.

In Figure 7 we see the 12 individuals of a run of Benford Law and GCF. In the Pareto front we see three clusters and one outlier on the lower right. Again we see a nice transition from one style to another; on the left we see two images in Benford Law style (we also see this type of images in Figure 1, left). Again we see images with high contrast and lot of black in the lower right of the Pareto front. Remarkable is the recurring ‘zebra’/ ‘tiger’ print motif that recurs in a number of images.

5 Conclusions and Future Work

Our first research question was “can we combine the effects from multiple aesthetic measures into the same image using a MOEA?”. The answer to this question is ‘Yes, but not necessarily with success’. We have seen that some combinations of aesthetic measure work better than others; some combinations of aesthetic measures result in images where the aesthetic properties do not blend very well. It suggests that it is very important to carefully select the aesthetic measures in a MOEA setup. Combinations of aesthetic measures with opposing goals (e.g. stark contrast vs. little contrast) will most likely not result in images with new or surprising results. Most likely, they will result in images where one property is dominant. However, it will not always be clear whether two aesthetic measures have opposing goals. Furthermore, in order to improve the artistic range of an evolutionary art system, it can be wise to use aesthetic measures that have “somewhat” different goals. So it seems that the most interesting combinations are of aesthetic measures that are different from each other but not too different.

Another strategy could be to use aesthetic measures that act on different dimensions of an image. For example, if one aesthetic measures focuses on texture, one focuses on a certain aspect of contrast and one focuses on composition aspects of the images, then the outcome of the different measures can be merged more efficiently. This strategy looks like an interesting direction to explore in future work.

The second research question was whether the use of MOEA would result in “better” images; we think that some combinations of aesthetic measures certainly result in more interesting images, whereby properties of both aesthetic measures are merged nicely. We also think that some combinations of aesthetic measures work counter-productive, and do not result in “better” images. Nevertheless, we think it can be a powerful tool for evolutionary art systems.

References

1. del Acebo, E., Sbert, M.: Benford’s law for natural and synthetic images. In: Neumann, L., et al. (eds.) [10], pp. 169–176
2. Bentley, P.J., Corne, D.W. (eds.): Creative Evolutionary Systems. Morgan Kaufmann, San Mateo (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
4. Greenfield, G.R.: Mathematical building blocks for evolving expressions. In: Sarhangi, R. (ed.) 2000 Bridges Conference Proceedings, pp. 61–70. Central Plain Book Manufacturing, Winfield (2000)

5. Greenfield, G.R.: Evolving aesthetic images using multiobjective optimization. In: Sarker, R., Reynolds, R., Abbass, H., Tan, K.C., McKay, B., Essam, D., Gedeon, T. (eds.) Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003, pp. 1903–1909. IEEE Press, Canberra (2003)
6. den Heijer, E., Eiben, A.: Comparing aesthetic measures for evolutionary art. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 311–320. Springer, Heidelberg (2010)
7. den Heijer, E., Eiben, A.: Using aesthetic measures to evolve art. In: IEEE Congress on Evolutionary Computation (CEC 2010), Barcelona (2010)
8. Matkovic, K., Neumann, L., Neumann, A., Psik, T., Purgathofer, W.: Global contrast factor-a new approach to image contrast. In: Neumann, L., et al. (eds.) [10], pp. 159–168
9. McCormack, J.: Open problems in evolutionary music and art. 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. 428–436. Springer, Heidelberg (2005)
10. Neumann, L., Sbert, M., Gooch, B., Purgathofer, W. (eds.): Computational Aesthetics 2005: Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging 2005, Girona, Spain, May 18-20. Eurographics Association (2005)
11. Romero, J., Machado, P. (eds.): The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Natural Computing Series. Springer, Heidelberg (2007)
12. Rooke, S.: Eons of genetically evolved algorithmic images. In: Bentley, P.J., Corne, D.W. (eds.) [2], pp. 339–365
13. Ross, B., Ralph, W., Zong., H.: Evolutionary image synthesis using a model of aesthetics. In: IEEE Congress on Evolutionary Computation (CEC 2006), pp. 1087–1094 (2006)
14. Ross, B.J., Zhu, H.: Procedural texture evolution using multi-objective optimization. New Gen. Comput. 22(3), 271–293 (2004)
15. Sims, K.: Artificial evolution for computer graphics. In: SIGGRAPH 1991: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, vol. 25, pp. 319–328. ACM Press, New York (1991)
16. Takagi, H.: Interactive evolutionary computation: Fusion of the capacities of ec optimization and human evaluation. Proceedings of the IEEE 89(9), 1275–1296 (2001)

A Genetic Algorithm for Dodecaphonic Compositions

Roberto De Prisco, Gianluca Zaccagnino, and Rocco Zaccagnino

Laboratorio di Musimatica
Dipartimento di Informatica
Università di Salerno
84084 Fisciano (SA) - Italy
http://music.dia.unisa.it

Abstract. In this paper we propose an automatic music composition system for dodecaphonic music based on a genetic algorithm.

Dodecaphonic music, introduced by A. Schoenberg, departs from the concept of tonality by considering all 12 notes equally important. Dodecaphonic compositions are constructed starting from a 12-note series, which is a sequence of all the 12 notes; the compositional process uses the starting 12-note series as a seed and builds the music creating new fragments of music obtained by transforming the seed series.

The algorithm proposed in this paper automates the compositional process taking a seed series as input and automatically creating a dodecaphonic composition. We have implemented the algorithm and we have run several tests to study its behaviour.

1 Introduction

In this paper we are interested in the design and implementation of an *algorithmic music composition system* for *dodecaphonic music*. The system must be capable of composing music without *any* human intervention.

Algorithmic music composition fascinates both computer scientists and musicians. The literature has plenty of works in this area, starting from the ILLIAC[1] suite by Hiller [7,8], and arriving to more recent efforts, like, for example, the ones by Cope [3] and Miranda [13].

Several music composition problems have been considered in the literature: 4-voice harmonizations, jazz solos, music styles reproduction, and others. To the best of our knowledge, no one has yet considered the possibility of using genetic algorithms for producing dodecaphonic compositions.

The dodecaphonic (or 12-note) technique has been introduced by Arnold Schoenberg in the 1920s: this technique departs from the concept of tonality, in which the notes of a given tonality are more important than notes outside the given tonality, and considers all the 12 notes equally important. Dodecaphonic music is built starting from a *seed series*: a particular sequence of the 12 different

[1] ILLIAC is the name of a computer built in the 50s.

© Springer-Verlag Berlin Heidelberg 2011

notes. An example of a 12-note series is: `[C,D,D♯,A♯,C♯,F,B,G♯,E,A,G,F♯]`. The series is not a theme but a source from which the composition is constructed. It can be transposed to begin on any of the 12 pitches, and it may appear in various derived forms.

Schoenberg dodecaphonic technique has been later generalized, by his disciples Berg and Webern and by others, to the serialism technique. In this paper however we consider only dodecaphonic music where the seed series is made up of all the 12 notes and the possible transformations are limited to certain types of series manipulation.

During the last few decades, algorithms for music composition have been proposed for several music problems. Various tools or techniques have been used: random numbers, formal grammars, cellular automata, fractals, neural networks, evolutionary algorithms, genetic music (DNA and protein-based music). The book by Miranda [12] contains a good survey of the most commonly used techniques.

In this paper we are interested in the use of genetic algorithms. Genetic algorithms are search heuristics that allow to tackle very large search spaces. A genetic algorithm looks for a "good" solution to the input problem by emulating the evolution of a population whose individuals are possible solutions to the given problem. Once we have provided an evaluation function for the compositions, composing music can be seen as a combinatorial problem in a tremendously large search space. This makes the genetic approach very effective for our problem. As we will explain later in the paper, in our algorithm each individual of the evolving population is a complete dodecaphonic composition created from the seed series.

We have implemented the algorithm using Java and we have run several tests. In the final section of the paper we report the results of the tests that show the behavior of the algorithm.

Related work. Several papers have proposed the use of genetic algorithms for music composition (e.g. [9,10,14,15]); to the best of our knowledge none of the papers that propose genetic algorithms has considered dodecaphonic music except for a recent paper by Maeda and Kajihara [11] that proposes a genetic algorithm for the choice of a seed series. The fitness function used exploits the concept of consonance and dissonance to evaluate the candidate solutions. The cited algorithm builds only the seed series of a dodecaphonic composition. Our genetic algorithm takes a seed series as input and produces a complete dodecaphonic composition.

Surfing the web, it is possible to find software that produces dodecaphonic compositions. However no details or documentation about the algorithms used is provided.

2 Music Background

In this section we briefly recall the needed musical background to understand the remainder of the paper.

The music system we are accustomed to is the 12-tone equal temperament which divides all pitches into octaves and within each octave defines 12 notes, which are `C,C♯,D,D♯,E,F,F♯,G,G♯,A,A♯,B`. The notion of "tonality" is the heart of the equal temperament. Roughly speaking, a tonality is a set of notes, which form a scale, upon which a composition is built. The 12 notes of the equal tempered system are "equally spaced" and this makes transposition between different tonalities very easy. A tonality is built starting from one of the 12 possible notes; hence we can have 12 different starting points (tonalities). For each tonality there are several modes, of which the *major* and *minor* modes are the most used ones.

A tonal composition has a main tonality, for example, `D` major, upon which the composition is built. Notes of that tonality, `D,E,F♯,G,A,B,C♯,D` in the example, are more important than other notes, and also specific degrees of the scale have special roles (e.g., tonic, dominant, sub-dominant).

Dodecaphonic music departs from the notion of tonality. We do not have anymore scales and tonalities but all 12 notes in the octave are equally important. There are no special degree functions for the notes in the scale; in fact there are no scales anymore. Instead of the scale the structural skeleton of the composition consist of a sequence (that is, a particular order) of the 12 notes called *seed series.* An example of 12-note series is provided in Figure 1. The figure emphasizes the intervals between the notes; such intervals are indicated with a number (the size) and a letter (the type). For example "2M" indicates a major second, while "2m" indicates a minor second; "A" stands for augmented and "P"for perfect. Notice that intervals are always computed as ascending; for example the interval between the fourth (A♯, or 10) and the fifth note (C♯, or 1) of the series is a minor third because we consider the first C♯ above the A♯.

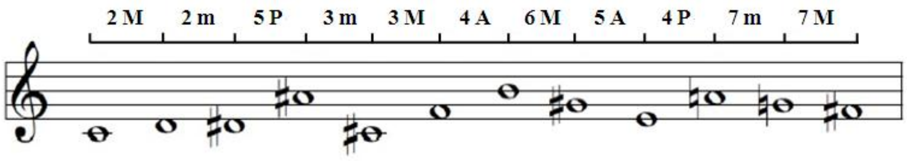


Fig. 1. Example of a seed series

A 12-note series is usually represented with an abbreviated form in which each of the twelve notes correspond to an integer in the range $[0, 11]$, where 0 denotes `C`, 1 denotes `C♯` , and so on up to 11 that denotes `B`. For example the series shown in Figure 1 corresponds to the sequence $[0, 2, 3, 10, 1, 5, 11, 8, 4, 9, 7, 6]$.

The seed series is then manipulated in order to obtain different fragments of music that are used to build up the dodecaphonic composition. Four standard manipulations can be used to obtain a set four of derived series, which are

1. `O`: Original, this is the seed series;
2. `R`: Retrograde, obtained by reversing the order of the intervals of the seed series;

3. `I`: Inversion, in which every interval in the seed series is inverted;
4. `RI`: Retrograde of Inversion, which is the retrograde of `I`.

Some theorists consider an extended set of derived series which includes beside `O, R, I, RI` also the following:

1. `D4`: Fourth, based on correspondences with the circle of fourths;
2. `D5`: Fifth, based on correspondences with the circle of fifths;
3. `R4`: Retrograde of `D4`;
4. `R5`: Retrograde of `D5`.

For more information about the derived series `D4` and `D5` see [6].

The derived series are not melodies, but only a reference schema for the composition. Rhythmic choices play a crucial role in creating the actual music from the derived series. Later we will explain how we choose rhythmic patterns. A *series transformation* will consists in the choice of a derived series and the application of a rhythmic pattern. Roughly speaking, the compositional process can be seen as the application of several series transformations.

The transformations can be used to obtain fragments of music for several voices to make up polyphonic compositions. Dodecaphonic music theory establishes some rules about the relative movements of the voices. We use such rules to evaluate the quality of the compositions.

3 The Algorithm

3.1 Rhythmic Patterns and Seed Series Transformations

The composition is made up of *transformations* of the seed series. A transformation of the seed series is obtained by first choosing a derived form of the seed series and then applying to such a derived form a rhythmic pattern. The derived forms that we consider are the ones explained in the previous section.

A rhythmic pattern P is a list $(n_1^{d_1,r_1}, \ldots, n_m^{d_m,r_m})$, with $m \geq 12$, where for each $n_k^{d_k,r_k}$ we have:

- $n_k \in \{0, 1, \ldots, 11, -1\}$, indicates the note ($-1$ indicates a rest note);
- $d_k \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}\}$, indicates the duration of note n_k;
- $r_k \in \{1, 2, \ldots, r_{max}\}$, indicates the number of repetitions of note n_k (r_{max} specifies the maximum number of repetitions that we allow).

Notice we have $m \geq 12$ because we can insert rests in the transformed series. For example, let F be the series $[0, 2, 3, 10, 1, 5, 11, 8, 4, 9, 7, 6]$ and P be the rhythmic pattern

$$(0^{\frac{1}{4},1}, 2^{\frac{1}{4},1}, 3^{\frac{1}{8},1}, 10^{\frac{1}{8},1}, -1^{\frac{1}{4},1}, 1^{\frac{1}{2},2}, 5^{\frac{1}{4},1}, 11^{\frac{1}{4},1}, 8^{\frac{1}{2},1}, 4^{\frac{1}{4},2}, -1^{\frac{1}{2},1}, 9^{\frac{1}{4},2}, 7^{\frac{1}{2},1}, 6^{1,1}),$$

where $m = 14$ because there are 2 rests. Figure 2 shows the result of a transformation of F obtained choosing the `R` (retrograde) derived form and applying the rhythmic pattern P to it. Remember that the intervals are always ascending intervals.

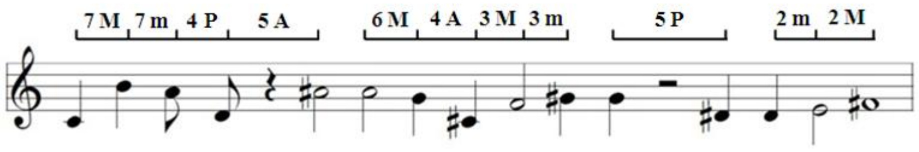


Fig. 2. Example of transformation

3.2 Chromosomes and Genes Representation

We represent each chromosome as an array C of dimension $K \times V$, where $K > 1$ denotes the number of melodic lines (voices) and V is an apriori fixed number that determines the length of the composition. We assume that all melodic lines are composed by the same number V of series transformations.

Formally, a chromosome C is an array $C = G_1, ..., G_V$, where each $G_i = T_i^1, ..., T_i^K$ is a gene and each T_i^j is a transformation of the seed series. Figure 3 shows a graphical representation of the structure of a chromosome. Notice that the blocks that represent genes are not aligned because the transformations T_i^j change the length of the seed series.

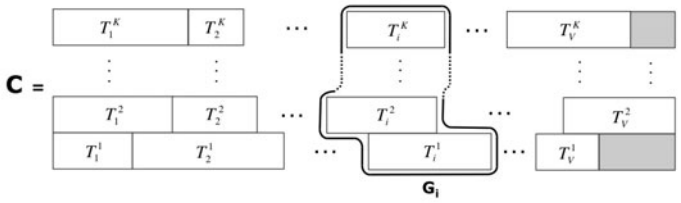


Fig. 3. Chromosomes and genes structure

3.3 Genotype-Phenotype Mapping

Since the seed series, and thus its transformations, does not specify the exact pitch of the notes (the same note can be played in several octaves, depending on the voice extension), we have that the chromosomes do not correspond immediately to solutions of the problem and thus our search space is different from the problem space. This implies that given a chromosome, in order to obtain a solution, we need to apply a genotype-phenotype mapping. In the following we define such a mapping.

To each voice we associate an *extension*, that is, a range of pitch values that the voice can reproduce. We use the MIDI values of the notes to specify the exact pitch. For example the extension $[48, 72]$ indicates the extension from $\texttt{C}_3$ through $\texttt{C}_5$. Recall that each note s in a series is denoted with an integer in the interval $[0, 11]$. We need to map such an integer s to a pitch (MIDI) value.

For each of the K voices, we fix a range $Ext_j = [\ell_j, u_j]$, $j = 1, 2, ..., K$, where ℓ_j is the lowest pitch and u_j is the highest pitch for voice j. Given a note $s \in [0, 11]$ played by voice j, the genotype-phenotype mapping has to choose an actual note by selecting a pitch in Ext_j.

The possible pitch values for a note s are the values in the set $Ext_j(s) = \{s' | s' \in Ext_j, s' \mod 12 = s\}$. For example, given $s = 0$, that is note `C`, and the extension $Ext = [57, 74]$, the possible MIDI values for s are $Ext(0) = \{60, 72\}$.

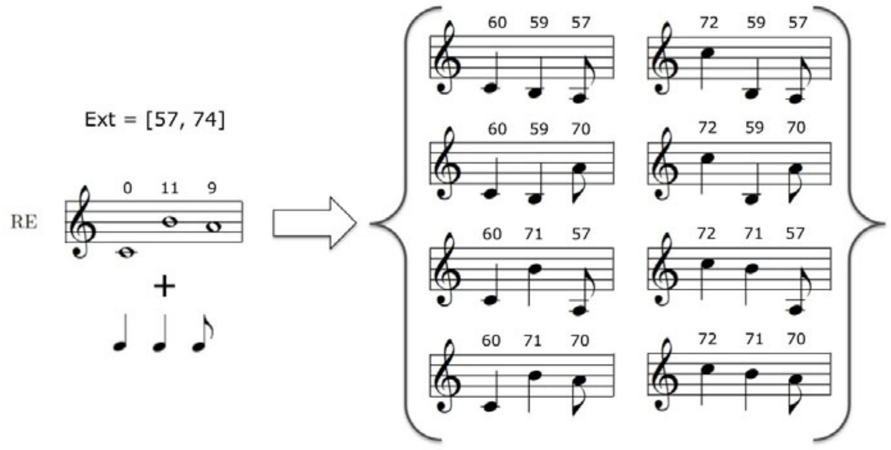


Fig. 4. An example of genotype-phenotype mapping

Figure 4 shows an example of genotype-phenotype mapping. To keep the example small we used a 3-note series instead of a complete 12-note series. In the example the chromosome consists of a single voice playing the 3-note series $\{0, 11, 9\}$, the voice extension is $Ext = [57, 74]$ and the rhythmic pattern is $(0^{\frac{1}{4},1}, 11^{\frac{1}{4},1}, 9^{\frac{1}{8},1})$. All the possible actual compositions that we can obtain in this simple example are 8 because for each of the 3 notes in the series we have exactly two possible actual pitches that we can choose. Figure 4 shows all these 8 possible compositions. Notice that with 12-note series and reasonable parameters K and V the space of all possible compositions is enormous.

The genotype-phenotype mapping has to choose one of the possible actual compositions to transform the chromosome in a solution of the problem. It does so by selecting uniformly at random one of the possible actual compositions. Notice that choosing an actual composition uniformly at random is equivalent to choosing uniformly at random an actual note for each abstract note.

We denote with *gpm* the genotype-phenotype mapping. Hence if we take a chromosome C and apply the genotype-phenotype mapping we obtain an actual music composition $X = gpm(C)$ where X is selected at random among all the possible compositions that correspond to C.

3.4 Initial Population

We start from a random initial population of N individuals. Each individual, which is made up of $K \times V$ seed series transformations, is constructed by selecting such transformations at random. The random choices are made as follows. For the derived form we choose at random one of the 8 forms explained in the music background section. In order to choose a random rhythmic pattern P for a derived series $[s_1, \ldots, s_{12}]$, we do the following

Let $i = 1$; do

1. Choose either
   (a) note s_i with an apriori fixed probability $p_{nr} > \frac{1}{2}$ or
   (b) -1 (rest) with probability $1 - p_{nr}$
   and let n_k be the result of the choice.
2. Assign to d_k a random value chosen in the set $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}\}$.
3. Assign to r_k a random value chosen in the set $\{1, 2, \ldots, r_{max}\}$.
4. Add $n_k^{d_k, r_k}$ to P.
5. If $n_k \geq 0$ then increment i (that is, if we have selected s_i and not a rest, proceed to s_{i+1}).
6. If $i = 13$ we are done. Otherwise repeat from 1.

3.5 Fitness Measure

In order to evaluate the current population and make a selection, we use rules taken from the dodecaphonic music theory. We refer the reader to textbooks such as [2,6] for an explanation of such rules. The specific set of rules that we have taken into account are detailed in Table 1. The rules refer to the movement of two voices of the composition. For example, the "parallel octave" rule says that two voices that proceed in parallel octaves are considered a critical error. As specified in the table, we have assigned a penalty of 100 to such an error. The "cross" rule says that two voices that cross each other are also considered an error, but not so critical. The penalty that we use in this case is lower.

Table 1. Errors and penalties for the evaluation function

Description	Penalty	Level
unison	30	normal
octave	30	normal
cross	30	normal
parallel fourth	40	normal

Description	Penalty	Level
parallel fifth	40	normal
parallel seventh	40	normal
parallel unison	100	critical
parallel octave	100	critical

We have taken the liberty of interpreting and adapting what is reported in dodecaphonic theory textbooks in order to obtain the rules reported in Table 1. For example voice crossing is normally allowed; however we do assign it a small penalty because we would like that voices do not cross too often. We have also classified these errors in two categories, normal and critical. The critical errors will be used for the mutation operator, as we will explain in later.

Let C be the chromosome and let $X = gpm(C)$ be the solution that will be used for the evaluation. Notice that this means that the actual fitness value that we use for C is the one that we obtain for X. In other words the fitness value is relative to the actual solution chosen by the *gpm* function.

Let d be the smallest note duration used in X. We will consider such a note duration as the "beat" of the composition. Notice that this "beat" is used only for the evaluation function. Depending on the value of d we can look at the entire composition as a sequence of a certain number z of beats of duration d. For each of these beats we consider all possible pairs of voices and for each pair we check whether an error occurs or not. If an error occurs then we give the corresponding penalty to X. The fitness value is obtained by adding all the penalties. The objective is to minimize the fitness value.

While evaluating the population, when we find a critical error we *mark* the corresponding gene. This information will be used in the mutation operator. Notice that if the error occurs across two genes we mark both genes.

3.6 Evolution Operators

In order to let the population evolve we apply a *mutation* operator and a *crossover* operator.

- **Mutation operator.** This operator creates new chromosomes starting from a chromosome of the current population. For each chromosome $C = G_1, ..., G_V$ in the current population, and for each gene $G_i = T_i^1, ..., T_i^K$, we check whether G_i has a critical error (that is if it has been marked in the evaluation phase) and if so we generate a new chromosome replacing G_i with a new gene $G'_i = {T'}_i^1, ..., {T'}_i^K$ where ${T'}_i^j$ is obtained by applying a transformation to T_i^j. The transformation is chosen at random. We remark that the transformation might include the "identity" in the sense that we might not apply a derivation and/or we might not change the rhythmic pattern.
- **Crossover.** Given two chromosomes C^1 and C^2, the operator selects an index $i \in \{1, \ldots, V-1\}$ randomly, and generates the chromosome $C^3 = G_1^1, \ldots, G_i^1, G_{i+1}^2, \ldots, G_V^2$.

3.7 Selection and Stopping Criteria

At this point as a candidate new population we have the initial population of N chromosomes, at most new N chromosomes obtained with the mutation operator and exactly $N(N-1)/2$ new chromosomes obtained with the crossover operator. Among these we choose the N chromosomes that have the best fitness evaluation. We stop the evolutionary process after a fixed number of generations (ranging up to 200 in the tests).

4 Test Results

We have implemented the genetic algorithm in Java; we used the JFugue library for the music manipulation subroutines. We have chosen as test cases several seed series taken from [6]. We have run several tests varying the parameters on many input series. The results obtained in all cases are quite similar.

Figure 5 shows the data obtained for a specific test using $K = 4$, $V = 30$, $p_{nr} = 0.8$. The size of the population is $N = 100$. Figure 5 shows the fitness value and the number of errors of the best solution as a function of the number of generations.

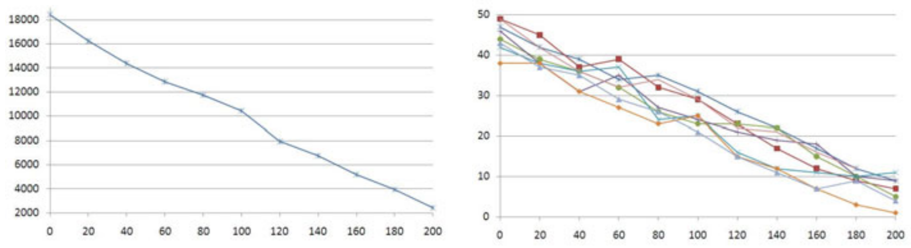


Fig. 5. Fitness value and errors as function of the number of generations

Since in this test we have used $V = 30$ the length of the composition doesn't allow us to show the best solution. However, just as an example, Figure 6 shows the actual composition given as output for a similar test in which we used $V = 2$ so that the total length of the composition is very small.

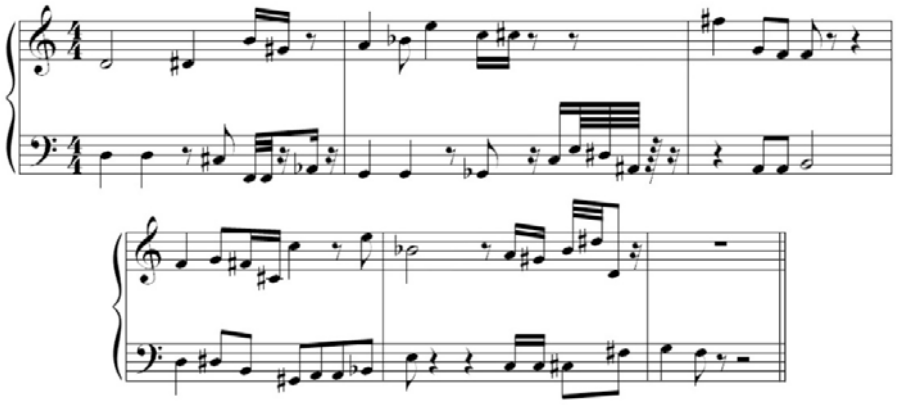


Fig. 6. Dodecaphonic music given as output for an example with $V = 2$

5 Conclusions

In this paper we have provided an automatic music composition system for dodecaphonic music using a genetic algorithm. The output of the system is promising. In this first version the system considers only the basic rules for the composition of dodecaphonic music. Future work might include: (a) study of the impact of the errors and penalties used; (b) use of a more complex fitness function taking into account not only harmony rules but also rhythmic and melodic considerations; (*c*) study of the impact of the random choices (for example the probabilities seed for the selection of the rhythmic pattern); (*d*) development of a more complex version of the system in order to include advanced features, like, for example, symmetry rules spanning the entire music composition.

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. Brindle, R.S.: Serial Composition. Oxford University Press, London (1966)
3. Cope, D.: Web page, `http://artsites.ucsc.edu/faculty/cope/`
4. De Prisco, R., Zaccagnino, R.: An Evolutionary Music Composer Algorithm for Bass Harmonization. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 567–572. Springer, Heidelberg (2009)
5. De Prisco, R., Zaccagnino, G., Zaccagnino, R.: EvoBassComposer: A multi-objective genetic algorithm for 4-voice compositions. In: Proceedings of the 12th ACM Conference on Genetic and Evolutionary Computation (GECCO), pp. 817–818. ACM Press, New York (2010)
6. Eimert, H.: Lehrbuch der Zwöfltontechnik. Breitkopf & Härtel, Wiesbaden (1950)
7. Hiller, L.: Computer music. Scientific American 201(6), 109–120 (1959)
8. Hiller, L., Isaacson, L.M.: Experimental music. McGraw-Hill, New York (1959)
9. Horner, A., Goldberg, D.E.: Genetic algorithms and computer assisted music composition. Technical report, University of Illinois (1991)
10. Horner, A., Ayers, L.: Harmonization of musical progressions with genetic algorithms. In: Proceedings of the International Computer Music Conference, pp. 483–484 (1995)
11. Maeda, Y., Kajihara, Y.: Automatic generation method of twelve tone row for musical composition using genetic algorithm. In: Proceedings of the IEEE 18th International conference on Fuzzy Systems FUZZY 2009, pp. 963–968 (2009)
12. Miranda, E.R.: Composing Music with Computers. Focal Press (2001)
13. Miranda, E.R.: Web page, `http://neuromusic.soc.plymouth.ac.uk/`
14. McIntyre, R.A.: Bach in a box: The evolution of four-part baroque harmony using a genetic algorithm. In: Proceedings of the 1st IEEE Conference on Evolutionary Computation, pp. 852–857 (1994)
15. Wiggins, G., Papadopoulos, G., Phon-Amnuaisuk, S., Tuson, A.: Evolutionary methods for musical composition. International Journal of Computing Anticipatory Systems 4 (1999)

A Customizable Recognizer for Orchestral Conducting Gestures Based on Neural Networks

Roberto De Prisco, Paolo Sabatino,
Gianluca Zaccagnino, and Rocco Zaccagnino

Laboratorio di Musimatica
Dipartimento di Informatica
Università di Salerno
84084 Fisciano (SA) - Italy
http://music.dia.unisa.it

Abstract. In this paper we present a system for the recognition of orchestral conducting gestures using the Nintendo Wiimote controller. The system is based on a neural network. This is not the first of such systems; however compared to previous systems that use the Wiimote or other cheap hardware, it has several advantages: it is fully customizable, continuous and does not require third party commercial software.

The system has been implemented in Java and we have run several tests to evaluate its behavior.

1 Introduction

Modern technologies allow the direct use of human gestures as a means of interaction with computers and other electronic devices. Hence the problem of gesture recognition has recently received considerable attention, both from a computer science point of view and from a linguistic point of view. The field is quite wide and includes recognition of facial expressions, hand gestures, body gestures and has a variety of applications in several fields.

In this paper we are interested in the application of gesture recognition to the music field and more in particular to the specific problem of recognizing orchestral conducting gestures. A number of systems for the recognition of orchestral conducting gestures have been developed. The list is quite long; here we provide only a few examples.

The Radio Baton [8] developed by Mathews is an electronic baton that uses radio signals to capture the movements of the baton. The conductor can control the tempo and the volume of the execution. The system requires specialized hardware for the recognition of the baton movements. The goal of the system is to "conduct" the execution (controlling tempo and dynamics) of a music piece played by an electronic orchestra made up of MIDI instruments.

Ilmonen and Takala [4] describe a system to extract information about rhythm and dynamics from the conductor gestures using a neural network. The system uses specialized hardware made up of magnetic sensors to track the conductor's movements.

© Springer-Verlag Berlin Heidelberg 2011

Kolesnik and Wanderlay [6] propose a system that captures conducting gestures using a pair of cameras, analyzes the images with the open software EyesWeb and feeds the resulting representation of the gestures to the Max/MSP commercial software. In the MAX/MSP environment the gestures are analyzed and recognized using the Hydden Markov Model.

The papers that we have cited above are only a few of the research efforts spent in order to have digital baton systems for musical conduction. The book by Miranda and Wanderley [9] is a great source of information about the acquisition and analysis of musical gestures and about the specific hardware that has been used in several projects.

Many previous systems use specialized hardware: special digital batons, cybergloves, special sensors, etc., which are not available as commercial products or can be very expensive to obtain or to build. In recent years the technology for human movement recognition, spurred by its application in videogames, has seen a rapid development. The result is the availability of commercial products like the Nintendo Wii console or the Kinetc Xbox. The Wii remote controller, often called Wiimote, is a very cheap hardware that allows to track user movements. A specific study, by Kiefer et. al. [3], has assessed the Wiimote usability for musical applications with respect to accuracy and user feedback provided by the Wiimote. There are a number of freely available Wiimote libraries for interfacing the Wiimote to a personal computer, e.g, [2], [14]. Many people have started using the Wiimote as the basic hardware for gesture recognition.

In this paper we propose a Wiimote-based system for the real-time (continuous) recognition of music conducting gestures. The proposed system is *continuous* in the sense that the user simply conducts the music without breaking down the entire sequence of gestures into separate single gestures: the system automatically analyzes the input and determines the separation of the single gestures. The proposed system is *customizable* in the sense that the specific set of recognized gestures is decided by the user that first has to train the system to learn the wanted gestures.

Related work. We found several papers in the literature that use the Wiimote for the recognition of musical gestures or for the recognition of more general gestures. The papers that tackle the specific problem of recognizing music conducting gestures are [1], [11] and [13].

The system built by Bradshaw and Ng [1] is a 3D tracking system of musical gestures. The goal of the system is the analysis and the 3D representation of the gestures. It is not meant for "understanding" the gestures, but only to provide a 3D representation. The system developed by Bruegge et al. [11] allows the tracking either with a Wiimote or with a camera. Few details about the gestures recognition are provided in [11]; it appears that what the Pinocchio system extracts from the user gestures are the indications about tempo and dynamics. The UBS Virtual Maestro developed by Marrin Nakra et al. [13] is similar to Pinocchio in that it also uses the Wiimote in order to conduct a virtual orchestra controlling the tempo and the dynamics of the execution. Both Pinocchio and UBS Virtual Maestro do not focus on the specific problem of recognizing orchestral music conducting gestures. The main focus of both systems is that of allowing the user to control tempo and dynamics of a virtual execution. It should be noted that fully recognizing conducting gestures is different from just understanding the tempo and the dynamics from the conducting gestures.

Among the projects that use the Wiimote for more general (not music related) gestures recognition we cite [12] because the system presented is enough general to be used also for music gestures recognition. Other papers, like [10] and [7] present specific recognizers which, at least from what we understand, cannot be used for continuous music gestures recognition.

The system presented by Schlömer et al. [12] is a general gesture recognizer that can be trained by the user. So it can be used for our problem. However for each gesture the user is required to push and hold the Wiimote button to indicate the start and the end of the gesture. This means that the recognition is discrete, that is, works for single gestures. This might be perfectly acceptable in many situations; for our problem, however, this means putting the burden of separating the gestures on the conductor, and thus it is not acceptable. Our system is continuous in the sense that the separation of the gestures is made automatically by the recognizer.

2 Background

2.1 Wiimote

The Wiimote controller, shown in Figure 1, produced and commercialized by Nintendo, is designed to allow user interaction with the Wii Console. The Wiimote can be used also as stand-alone hardware and can be connected to a personal computer through Bluetooth. Several open source libraries allow to read the data produced by the Wiimote and provide a representation of the user movement as a list of points. In our particular setting, we have used an infrared pen as digital baton (see Figure 1); the Wiimote controller captures the movement of the pen and sends the data to the personal computer. Both the Wiimote controller and the infrared pen are very cheap. Notice that we do not need the Nintendo Wii console but only the remote controller.

Fig. 1. The Wiimote controller (left) and infrared pens (right)

The data arrives as the list of point coordinates of the pen relative to a reference box defined in the Cartesian space. Such a reference box has to be defined in a calibration phase. There are a number of open source projects that show how to implement the above system, e.g. [2]. An example of input is $(100, 100), (100, 101), (103, 105), (102, 102), (98, 101), (97, 96), (90, 92), (80, 88), ...$

2.2 Feed-Forward Neural Networks Classifier

Neural networks are widely used to reproduce some activities of the human brain, for example, perception of images, pattern recognition and language understanding. They are a very useful tool also for gestures recognition. One of the most commonly used neural network is the fully connected three-layer feed-forward neural network. The neurons activation function is a sigmoid function defined as $f(a) = \frac{1}{1+exp(-a)}$. We do use such a kind of neural network.

Our problem can be seen as a *classification problem* in which the network has to be able to recognize gestures and classify them in one of k classes $C_1, C_2, ..., C_k$, that can be decided by the user.

2.3 Discrete vs. Continuous Recognition

An important distinction that we have to make is about discrete (or static) and continuous (or dynamic) recognition. A discrete recognition means that we assume that the input is just one single gesture, that is we can recognize one gesture at a time. If we want to recognize a sequence of gestures with a discrete recognizer we need to break down the list that represents all the gestures into pieces, each of which corresponds to a single gesture.

Conversely, a continuous recognition is able to take as input the complete list that represents all the gestures and automatically recognize each single gesture, which implies being able to automatically break down the input sequence into the pieces corresponding to single gestures.

Clearly, continuous gestures recognition is much harder than discrete gesture recognition. One way of breaking down the complete list of the gestures into fragments for each single gesture is to put the burden on the conductor: the electronic baton might have a button that the conductor has to press at the beginning of each gesture. However this is not a natural thing to do for the conductor. It is more realistic to assume that the electronic baton will provide a list of point coordinates that describes all the gestures without explicit indications about when the previous gesture ends and the subsequent one begins.

2.4 Turning Points

A turning point of a gesture is a change of direction of the gesture. A simple way to define turning points is the following. Look at the x and the y coordinates and as long as they proceed in ascending or descending order we say that the direction has not changed. When one of the coordinates changes its direction then we have a turning point. For example if the list of coordinates that describes a gesture $(100, 100), (99, 101), (98, 102), (99, 103), (104, 110), (107, 109), ...$, we have 2 turning points, one between the third and the fourth point and another one between the fifth and the sixth point.

3 The Recognizer

In this section we describe the neural network for the recognition of music conducting gestures. We use a feed-forward three-layer neural network. Our goal is to recognize typical music conducting gestures. Such gestures provide indications to the musicians mainly about the tempo and the dynamics to be used. However specific gestures might have also other meanings to the musicians. The gesture meanings are decided by the conductor.

Each gesture has peculiar shape and characteristics. Figure 2 shows the 6 standard gestures that we have used for the tests. We remark that the system is customizable so that the set of recognized gestures can be decided by the user. For this particular set we have $k = 6$ gestures and the number of turning points are $n_1 = 8$, $n_2 = 10$, $n_3 = 12$, $n_4 = 14$, $n_5 = 16$, $n_6 = 18$ and thus $n_{\max} = 18$.

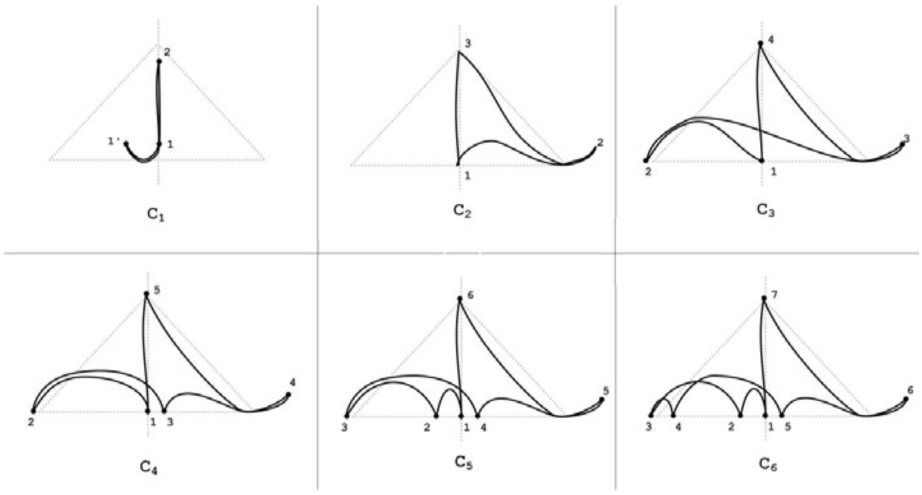


Fig. 2. The set of 6 conducting gestures classes $C_1, C_2, ..., C_6$ used as a test case

As we have said in the background section our input is provided by the Wimote controller and is a list G of point coordinates:

$$G = (x_1, y_1), (x_2, y_2,), \ldots\ldots\ldots, (x_i, y_i), (x_{i+1}, y_{i+1}), \ldots\ldots$$

Each pair is a point in a reference rectangle with corners at $(0, 0)$ and $(1024, 768)$ in the Cartesian space. The particular values 1024 and 768 are specific to the resolution of the Wiimote controller.

As in many other classification problems we use a data filtering to "smooth" the input data and make the recognition problem easier. The raw input data for a gesture G acquired using the Wiimote controller might be not a perfect representation of the intended gesture. This might be due to the fact that the gesture is not always the same, to the approximation introduced by the Wiimote or to any other source of noise in the raw data.

3.1 Data Representations

In this section we describe the data representation for the input and the output of the neural network. We remark that the choice of the data representation is a crucial step in the design of a neural network. The output of the network heavily depends on such a choice. We experimented 4 alternative data representations and we have compared the output of each type to determine the best data representation among the proposed 4 alternatives.

Representation 1. Given a gesture G (input), the corresponding filtered data F, let d be the distance between the two farthest turning points, P_A and P_B, in F and let r be the straight line passing through P_A and P_B. Notice that P_A and P_B do not necessarily correspond to the first P_1 and the last P_n turning points; actually they are different in many cases.

For each turning point P_i of a gesture we consider the perpendicular projection X_i of P_i on r and include in the data representation the following information:

– the normalized height $\frac{h_i}{d}$ of P_i; the height is positive for points on one half-plane of r and negative for points on the other half-plane;
– the normalized distance $\frac{d_i}{d}$ between X_i and P_A;
– the normalized distance $\frac{d-d_i}{d}$ between X_i and P_B.

Observe that each value is a real number in $[-1, 1]$.

Hence a gesture F is represented with a list of triples $F \simeq ((\frac{h_1}{d}, \frac{d_1}{d}, \frac{d-d_1}{d}), ..., (\frac{h_n}{d}, \frac{d_n}{d}, \frac{d-d_n}{d}))$. Notice that the value of n might change depending on the particular gesture. Remember that $n_{\max}$ is the maximum number of turning points in any gesture.

The first-layer of the network consists of $N_{in} = 3n_{\max}$ neurons to which the above representation (with a possible pad of zeroes) is fed. The output of the network is a string of k bits, where k is the number of different gestures (classes) that we have to recognize. Each class corresponds to a sequence of bits consisting of $k-1$ zeroes and exactly a 1. Class C_1 is codified with $[1, 0, \ldots, 0]$, C_2 is codified with $[0, 1, \ldots, 0]$ and so on. Hence the third layer of the network consists of $N_{out} = k$ neurons.

Representation 2. In the second representation we use as reference line the straight line r passing through the first and the last turning points, that is P_1 and P_n. For each turning point P_i we consider the straight line r_i passing from P_1 and P_i and represent P_i with the radian of the angle between r and r_i.

We represent the gesture F with the list of radians $F \simeq (r_1, ..., r_n)$. As before, the value of n might change depending on the particular gesture. The first layer of the network consists of $N_{in} = n_{\max}$ neurons to which the above representation (possibly padded with zeroes) is fed. As for Representation 1, the output is specified with a string of k bits, hence $N_{out} = k$.

Representation 3. In the third representation the reference line r depends only on P_1 and is the straight line passing from P_1 and parallel to the horizontal axis of the Cartesian reference system. The data representation is similar to the one used in Representation 2: gesture F is represented with the list of radians $F \simeq (r_1, ..., r_n)$ where r_i is the angle measured in radians between the reference line r and the straight line passing through P_1 and P_i. The first layer of the network consists of $N_{in} = n_{\max}$ neurons to which the above representation (possibly padded with zeroes) is fed. As for the previous representations, the output is specified with a string of k bits, hence $N_{out} = k$.

Representation 4. In the fourth representation instead of a reference line we use a reference rectangle: we consider the smallest rectangle of the Cartesian space that includes all the turning points. The lower-left corner of such a rectangle is point A with coordinates $(x_{\min}, y_{\min})$ where $x_{\min} = \min x_i$ over all the turning points $P_i = (x_i, y_i)$ and $y_{\min} = \min y_i$ over all the turning points P_i. The upper-right corner of the rectangle is point B with coordinates $(x_{\max}, y_{\max})$ where $x_{\max} = \max x_i$ and $y_{\max} = \max y_i$. Let d be the distance between A and B. For each turning point P_i we include in the data representation the following information about P_i:

– the value of $\cos \alpha_i$ where α_i is the angle between the straight line passing by A and P_i and the horizontal Cartesian axis;
– the normalized value $\frac{d_i}{d}$, where d_i is the distance between A and P_i.

We represent the gesture F with the list of pairs $F \simeq ((\cos \alpha_1, \frac{d_i}{d}), ...(\cos \alpha_n, \frac{d_n}{d}))$. The first-layer of the network consists of $N_{in} = 2n_{\max}$ neurons to which the above representation (with a possible pad of zeroes) is fed. For this representation we adopt a new strategy about the output representation: the output will be specified using the same representation used for the input. Hence $N_{out} = 2n_{\max}$.

The motivation for this choice is that we wanted to get a better behavior for the continuous recognition and requiring the network to provide an enumerative output, as in the previous representations, makes the task of the network harder. When the network is required to give an output in the same form of the input the task of the recognition becomes easier.

3.2 Continuous Recognition

As we have said earlier, the continuous recognition of gestures is much harder than the discrete one because we have a continuous list of point coordinates and we do not know when a gesture ends and consequently when a new gesture starts. In order to tackle this problem we define the *gesture fragments*. A gesture fragment is a "piece" of the gesture. We can define fragments exploiting the turning points. In its simplest form a fragment is simply the piece of a gesture between two successive turning points. However a fragment might consists also of 2 or more consecutive turning points. In general, for a given network we fix a parameter f, the number of fragments, and split each gesture in at most f fragments as follows. Let n be the number of turning points in the gesture. If $n \geq f + 1$ then the gesture will be divided into f fragments each consisting of either $\lfloor n/f \rfloor$ or $\lceil n/f \rceil$ consecutive turning points; if $n \leq f$ then the gesture will be split into $n-1$ fragments each one consisting of two consecutive turning points.

In the training phase, in which the user trains the system performing sample gestures, the system automatically computes fragments and learns how to recognize the single fragments.

The continuous recognition works as follows. We fix a sample time interval t_S. We look at the input every t_S units of time. The shorter t_S is the better is be the recognition; clearly we cannot look at the input too frequently because we are bound to the CPU requirements of the neural network. In practice t_S will be the smallest time unit that we can use. In order to dynamically recognize gestures, we feed the neural network with the input at time t_S (the input starts arriving at time 0). The neural network will respond saying either that it recognizes in the input a particular class gesture C_i or a particular fragment of a gesture. If the output of the network is not a gesture then we define the current output error to be 1. If the output is a gesture C_i we first compute the difference vector between the input data and C_i and then we define the error to be the norm of the difference vector. Notice that the error is always in $[0, 1]$. If the error is above a pre-defined threshold T then it is likely that the recognized gesture has been misclassified; for example because the system has recognized only some fragments. In this case we wait another time unit t_S and we repeat the recognition but using the input data from 0 through the current time until the error distance is below the threshold; when this happens the network has recognized one complete single gesture. This allows to split the overall list of gestures into separates pieces, each one belonging to one single gesture.

Time 0 is to be interpreted as the beginning of the new gesture; that is when we recognize a gesture we reset the time to 0 for the recognition of the subsequent gesture.

The choice of the threshold T affects the quality of the recognition. After several attempts we have set this threshold to $T = 0.075$ because with this value we obtained better results.

4 Training, Validation and Test Results

Discrete recognition. Our training set consisted of 100 repetitions of each single type of the 6 gestures for a total of 600 training gestures performed by a single user. The validation set and the test set were obtained similarly. The validation set consisted of 230 gestures (a little less than 40 per each type) and the test set consisted of 125 gestures (about 20 per type).

We have run tests for each of the 4 data representations described in the previous section. All networks were trained for about 5000 epochs. The error rates for the 4 representations were, respectively, 9.9%, 11.8%, 11.4%, 4.1%. Hence the best network is the one based on Representation 4.

Continuous recognition. In order to test the behavior of the network for continuous gesture recognition we have run several tests using sequences of various lengths, ranging from sequences of 10 consecutive gestures to sequences of 100 consecutive gestures. For each sequence we have repeated the test 10 times. Table 1 provides the percentage of errors for each of the 10 tests performed for the case of the sequence of 20 gestures. Table 2 provides the average error (over 10 tests) for each set of tests for the various sequences of 10 through 100 gestures. As can be seen from the results, the neural network that uses Representation 4 provides the best results. It is worth noticing that the behavior of all the networks degrades as the length of the gesture sequence increases. This might be due to some error propagation.

Table 1. Error rates for the test with a sequence of 20 gestures

Test number	1	2	3	4	5	6	7	8	9	10	Average
R1	40%	33%	38%	37%	30 %	38%	35%	35%	31%	35%	35.2%
R2	28%	31%	22%	22%	25%	27%	26%	22%	22%	27%	25.2%
R3	20%	29%	28%	28%	22%	19%	25%	25%	25%	26%	24.7%
R4	2%	3%	5%	4%	7%	7%	9%	3%	2%	2%	4.4%

Table 2. Average error rates for sequences of gestures with various length (from 10 through 100)

Sequence length	10	20	30	40	50	60	70	80	90	100
R1	33%	35%	40%	40%	46%	48%	54%	56%	28%	70%
R2	21%	25%	28%	31%	31%	31%	37%	39%	42%	44%
R3	20%	25%	26%	28%	31%	28%	35%	38%	42%	45%
R4	2%	4%	5%	4%	7%	7%	9%	11%	13%	16%

5 Conclusions and Future Work

Possible directions for future work can be two-fold. On one hand one could study possible improvements of the system in terms of error rates by fine-tuning the system, exploring other data representation or even different approaches for the continuous recognition. On the other hand, one can integrate the recognition system into a bigger framework, for example, for orchestral score following or for the integration of a particular director's conducting gestures into the score.

References

1. Bradshaw, D., Ng, K.: Tracking conductors hand movements using multiple wiimotes. In: Proceedings of the International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution (AXMEDIS 2008), Firenze, Italy, pp. 93–99. IEEE Computer Society Press, Los Alamitos (2008)
2. Lee, J.C.: Hacking the Nintendo Wii remote. Pervasive Computing 7(3), 39–45 (2008)
3. Kiefer, C., Collins, N., Fitzpatrick, G.: Evaluating the Wiimote as a musical controller. In: Proceedings of the International Computer Music Conference, Belfast, N. Ireland (2008)
4. Ilmonen, T., Takala, T.: Conductor following with artificial neural networks. In: Proceedings of the International Computer Music Conference, Bejing, China (1999)

5. Hofmann, F.G., Heyer, P., Hommel, G.: Velocity profile based recognition of dynamic gestures with discrete Hidden Markov Models. In: Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction, pp. 81–95. Springer, London (1998)
6. Kolesnik, P., Wanderley, M.: Recognition, analysis and performance with expressive conducting gesture. In: Proceedings of the International Computer Music Conference, Miami, USA (2004)
7. Liang, H., Ouhyoung, M.: A real-time continuous gesture recognition system for sign language. In: Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition, p. 558 (1998)
8. Mathews, M.: The radio baton and the conductor program, or: Pitch-the most important and least expressive part of music. Computer Music Journal 15(4), 37–46 (1991)
9. Miranda, E.R., Wanderley, M.: New digital musical instruments: Control and interaction beyond the keyboard. A-R Editions, Middleton (2006)
10. Mlich, J.: Wiimote gesture recognition. In: Proceedings of the 15th Conference and Competition STUDENT EEICT 2009, vol. 4, pp. 344–349 (2009)
11. Bruegge, B., Teschner, C., Lachenmaier, P., Fenzl, E., Schmidt, D., Bierbaum, S.: Pinocchio: Conducting a virtual symphony orchestra. In: Proceedings of the International Conference on Advances in Computer Entertainment Technology (ACE), Salzburg, Austria, pp. 294–295 (2007)
12. Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a Wii controller. In: Proceedings of the 2nd International ACM Conference on Tangible and Embedded Interaction, New York, NY, USA, pp. 11–14 (2008)
13. Nakra, T.M., Ivanov, Y., Smaragdis, P., Ault, C.: The UBS virtual maestro: An interactive conducting system. In: Proceeding of 9th International Conference on New Interfaces for Musical Expression, Pittsburgh, PA, USA (2009)
14. Wii yourself website, http://wiiyourself.gl.tter.org/

Generative Art Inspired by Nature, Using NodeBox

Tom De Smedt[1,2], Ludivine Lechat[2], and Walter Daelemans[1]

[1] Computational Linguistics & Psycholinguistics Research Group,
University of Antwerp, Belgium
[2] Experimental Media Group, Sint Lucas
Antwerpen, Belgium
tom@organisms.be, ludivinelechat@gmail.com,
walter.daelemans@ua.ac.be

Abstract. NodeBox is a free application for producing generative art. This paper gives an overview of the nature-inspired functionality in NodeBox and the artworks we created using it. We demonstrate how it can be used for evolutionary computation in the context of computer games and art, and discuss some of our recent research with the aim to simulate (artistic) brainstorming using language processing techniques and semantic networks.

Keywords: computer graphics, generative art, emergence, natural language processing.

1 NodeBox

1.1 Computer Graphics and User Interfaces

Traditionally, user interfaces in computer graphics applications have been based on real-world analogies (e.g., a pen for drawing, scissors for slicing). This model raises creative limitations. First, the features can only be used as the software developers implemented them; creative recombination of tools is impossible when not foreseen. Second, there is little room for abstraction: users will tend to think along the lines of what is possible with the built-in features (buttons, sliders, menus), and not about what they want [5].

In 2002 we released NodeBox[1], a free computer graphics application that creates 2D visual output based on Python programming code, with the aim to overcome these limitations. By writing Python scripts, users are free to combine any kind of functionality to produce visual output. This approach has also been explored in software applications such as Processing [19] (using Java code) and ContextFree (using a context-free grammar). Over the course of two research projects the application has been enriched with functionality for a variety of tasks, bundled in intermixable Python modules—for example, for image compositing, color theory, layout systems, database management, web mining and natural language processing.

[1] NodeBox for Mac OS X version 1.9.5, http://nodebox.net

© Springer-Verlag Berlin Heidelberg 2011

Example script. Images are downloaded using the Web module and arranged in a random composition, using the NodeBox rotate() and image() commands.

```
import web
images = web.flickr.search("flower")
for i in range(100):
    img = choice(images).download()
    rotate(random(360))
    image(img,
        x=random(800),
        y=random(600), width=200, height=200)
```

A number of modules are inspired by nature. For example, the Graph module combines graph theory (i.e., shortest paths, centrality, clustering) with a force-based physics algorithm for network visualization. The Supershape module implements the superformula [10], which can be used to render (and interpolate between) many complex shapes found in nature (ellipses, leaves, flowers, etc.). The L-system module offers a formal grammar that can be used to model (the growth of) plants and trees [18]. The Noise module implements Perlin's pseudo-random generator, where successive numbers describe a smooth gradient curve [17]. This technique is used in computer graphics to generate terrain maps, clouds, smoke, etc. Finally, two modules provide functionality for working with agent-based AI systems. The Ants module can be used to model self-organizing ant colonies. The Boids module presents a distributed model for flocking and swarming [20]. "Boids" is an emergent Artificial Life program where complexity arises from the interaction between individual agents. Each boid will 1) steer away to avoid crowding other boids, 2) steer in the average direction of other boids and 3) steer towards the average position of other boids.

1.2 Generative Art

In practice, NodeBox is used to create what is called "generative art". Generative art is an artistic field inspired by ideas about emergence and self-organization, and making use of techniques borrowed from AI and artificial life [2, 14]. The concept of emergence was first coined by Lewes (1875) and later described by Goldstein (1999) as "the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems" [11]. In terms of generative art, emergence implies that the artist describes the basic rules and constraints, and that the resulting artwork is allowed a certain amount of freedom within these constraints to self-organize.

In this sense NodeBox is for example useful for: a graphic designer producing a 200-page document in one consistent visual style but with variations across pages, information graphics based on real-time data, evolutionary art installations that react to input (e.g., sound), customized wallpaper based on e-mail spam [16], and so on. In section 2 we discuss one such project, which demonstrates how the software can be used for evolutionary computation in the context of the visual arts. In section 3 we show three example works of generative art.

An approach using programming code leads to new opportunities, but it also introduces a problem: many people active in the arts (e.g., art students) are not trained in programming. In section 4 we briefly discuss our attempts to alleviate this problem with a natural language processing approach, and by using a node-based interface.

2 Evolutionary Computation in NodeBox

2.1 Genetic Algorithms and Swarming

In 2007 we created Evolution,[2] a NodeBox art installation based on boid swarming and a genetic algorithm (GA). A starting set of creatures is randomly designed from a pool of components – heads, legs, wings, tails, etc. Different components have a behavioral impact. For example: the type of head allows a creature to employ better hunting strategies (ambush, intercept), better evasive strategies (deception, hide in the flock), or better cooperative skills. Larger wings allow a creature to fly faster.

Central in a GA's design is the fitness function, which selects optimal candidates from the population for the next generation. Here, the fitness function is an interactive hunting ground where creatures are pitted against each other. Survivors are then recombined and evolved into new creatures. Evolution's GA uses a Hierarchical Fair Competition model (HFC) [12]. HFC ensures that a population does not converge into a local optimal solution too quickly, by ensuring a constant supply of new genetic material (i.e., new random creatures to fight). Interestingly, when correctly tweaked this produces an endless crash-and-bloom cycle of 1) creatures that are exceptional but flawed and 2) mediocre all-rounders. Random newcomers will eventually beat the current (mediocre) winner with an "exceptional trick" (e.g., very aggressive + very fast), but are in turn too unstable to survive over a longer period (e.g., inability to cope with cooperating adversaries). Their trick enters the gene pool but is dominated by generations of older DNA, leading to a very slow overall evolution.

2.2 City in a Bottle – A Computer Game on Evolution by Natural Selection

Later, we expanded this prototype into a computer game project (City in a Bottle) based on the principles of emergence and evolution by natural selection. The project is currently in development. In short, the game environment is procedural, i.e., lacking a predefined landscape or storyline. Organisms (plants and insects) are described in terms of their basic behavioral rules: *"if attacked, flee"*, *"when cornered, fight"*. Complex game mechanisms then arise as organisms interact. If the most nutritious food is found in tall-stemmed flowers, creatures with wings will thrive—and in turn the spores from this kind of flower will spread. The game mechanisms are inspired by complex systems [13]: neither the designers nor the players of the game control the environment in full; only local actions such as planting a seed or catching and domesticating an insect are allowed.

The game music adheres to the same principle. Typically, computer games use a predefined library of sound effects that accompany an event (a weapon that is fired, a spell that is cast) and music tracks that are looped in specific situations (the haunted mansion music, the magic forest music). However, audio in City In A Bottle is composed in real-time, resulting in an emergent, swarm-based music score [1] where individual audio samples are composed based on the creature's wing flap velocity, the

[2] Evolution prototype, `http://nodebox.net/code/index.php/Evolution`

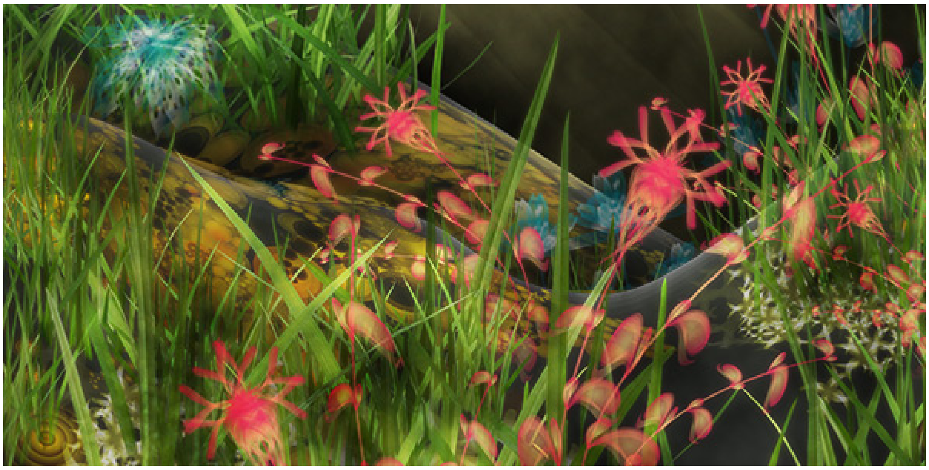


Fig. 1. Prototype of the City In A Bottle game world, with two kinds of flowers thriving

clicking of mandibles and the rushing of leaves. The overall music composition then arises as an interaction of creatures flocking together near food or wind rustling the leaves of various plants.

On a final note, the project uses a spin-off of NodeBox called NodeBox for OpenGL,[3] which uses hardware-acceleration on the computer graphics card for better performance.

3 Examples of Generative Art Created with NodeBox

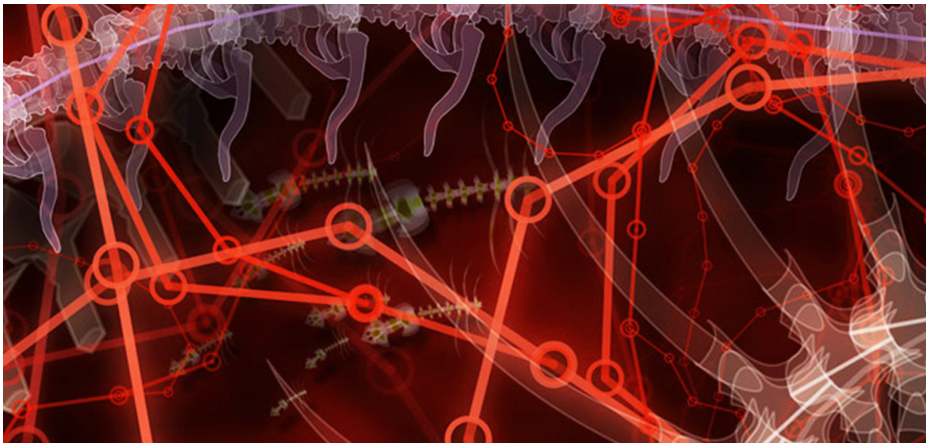


Fig. 2. "Creature": 350x150cm panel created for the department of Morphology, University of Ghent. It was realized using a recursive approach to simulate veins and skeleton structures.

[3] NodeBox for OpenGL, version 1.5, `http://cityinabottle.org/nodebox`

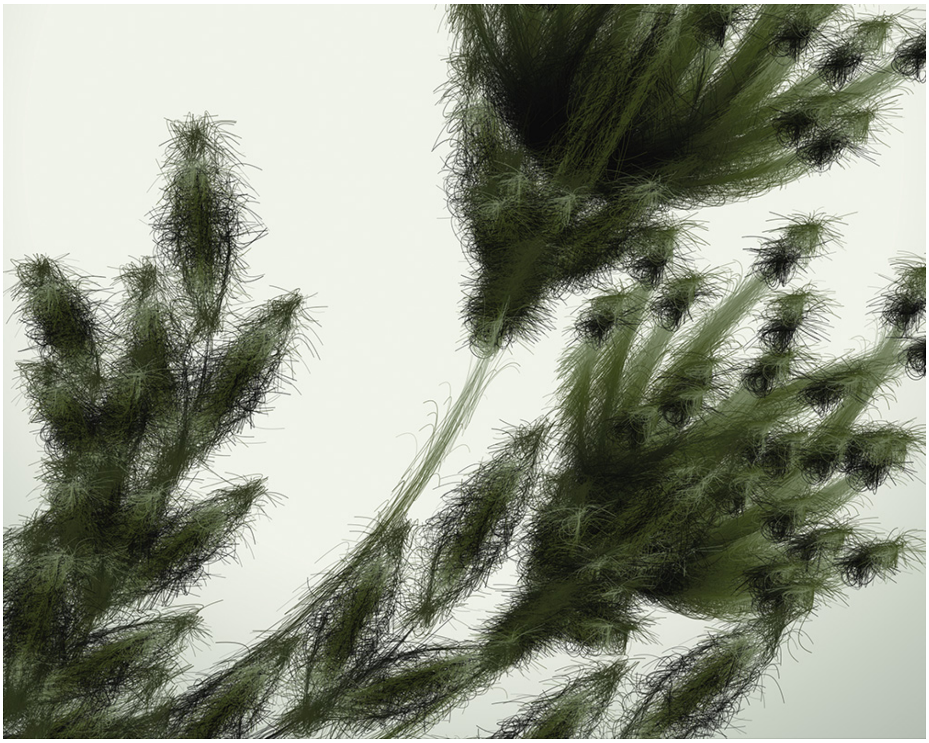


Fig. 3. "Superfolia": 6 panels 70x150cm realized using an agent-based approach (a single blade of grass responds to its neighbors)

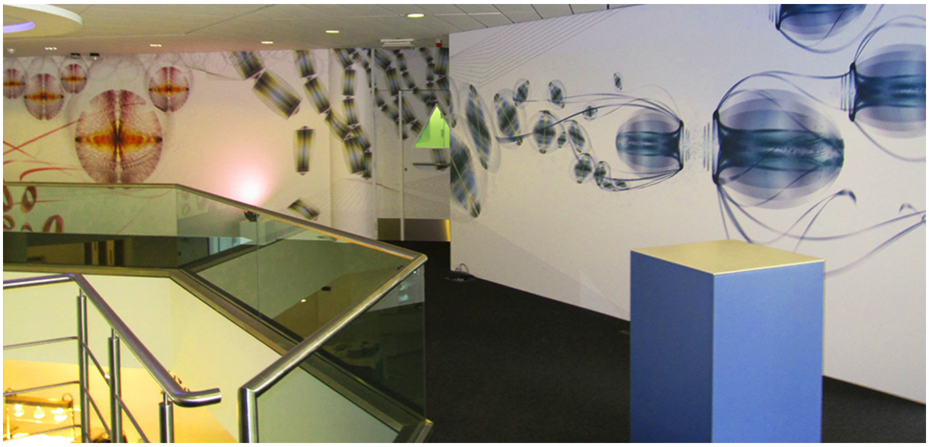


Fig. 4. "Nanophysical": 66.5x2.5m wall design at IMEC (European institute for nanotechnology). The work regards the hall window as a source of energy and then evolves along the walls, using (among other) a force-based physics algorithm.

4 Computational Creativity

A user interface with programming code introduces a steep learning curve for users not trained in programming. In a recent research project ("Gravital"), we have attempted to alleviate this shortcoming by providing a node-based interface.[4] Visual building blocks (nodes) can be connected in the interface to create interesting visual effects. Building blocks can be opened to examine and edit their source code.

Furthermore, a programming tool for the visual arts is useful in terms of production-intensive tasks, but it does not provide leverage on *what* to make—what ideas are "interesting" from a creative standpoint. The second aim in the Gravital project was to develop a set of algorithms to find creative associations and analogies between concepts (i.e., words), to help users discover interesting new ideas. The system uses a memory-based shallow parser [8], a semantic network of commonsense [22] and heuristic search techniques. We hypothesize that this system can be used to simulate conceptual brainstorms based on natural language input.

4.1 Memory-Based Shallow Parser

The first task in the system is to transform information in natural language sentences to a meaning representation language. This task has a long history in AI, and in practice the translation of natural language into a deep, unambiguous representation (i.e., understanding) turned out to be impossible (except for small domains where all relevant background knowledge was explicitly modeled, see for example [23]). Natural language processing (NLP) has since switched to robust, efficient and reasonably accurate methods that analyze text to a more superficial partially syntactic and partially semantic representation (shallow parsing), using machine learning and statistical methods trained on large annotated corpora.

The shallow parser used by our system is MBSP; a memory-based shallow parser implemented as memory-based learning modules using the Machine Learning package TiMBL [7]. Memory-based learning is a form of exemplar-based learning that is based on the idea that language processing involves specific exemplars of language use, stored in memory. With MBSP we can process user input in the form of natural language (i.e., English) and mine relevant concepts from it. These are then processed further with the Perception[5] solver: a semantic network traversed with heuristic search techniques.

4.2 Semantic Network of Commonsense

For example, assume we have a drawing machine that can draw either circles or rectangles, in any color. The task "draw a circle" is trivial and can be solved by the user himself without having to rely on NLP algorithms. The task "don't draw anything except an ellipse preferably of equal width and height" is quite complex to solve in terms of NLP, and perhaps not worth the effort. However: "draw the sun" poses an interesting challenge. What does the sun look like? Given the possibilities of

[4] NodeBox 2, beta version, `http://beta.nodebox.net/`
[5] Perception module, beta version,
`http://nodebox.net/code/index.php/Perception`

our drawing machine, a human might translate the "sun" concept to an orange circle. This kind of conceptual association is a form of human creativity [15], which we attempt to simulate using a semantic network of related concepts. When given the word "sun", Perception will propose colors such as orange and yellow, and shapes such as a circle or a star.

To illustrate this further, say we are looking for images of creepy animals. The system could search the web for images named `creepy-animal.jpg`, but that is not very creative. What we want is a system that imitates an artistic brainstorming process: thinking about what animals look like, what the properties of each animal are, which of these properties can be regarded as creepy, and look for pictures of those animals. In this particular example the Perception solver suggests such animals as octopus, bat, crow, locust, mayfly, termite, tick, toad, spider, ... No frolicking ponies or fluffy bunnies here! For the octopus the logic is obvious: the semantic network has a direct *creepy is-property-of octopus* relation. The bat (second result) has no *is-creepy* relation however, only a set of relations to *black*, *cave*, *night* and *radar*. What happens here is that many aspects of a bat are inferred as a strong causal chain [21] leading to creepiness. Let us clarify the meaning of "many aspects".

In [4], Hofstadter argues that AI-representations of human high-level perception require a degree of flexibility (or fluidity), where objects and situations can be comprehended in many different ways, depending on the context. To reflect this, Perception's solver uses clusters of concepts as its basic unit for reasoning, instead of a single concept. Concepts are surrounded by other concepts that reinforce meaning. A concept cluster is the concept itself, its directly related concepts, concepts related to those concepts, and so on, as deep as the representation requires (we used depth 2). This is called spreading activation [6]. Activation spreads out from the starting concept in a gradient of decreasing relatedness. What defines the bat concept are its directly surrounding concepts: *black*, *cave*, *night*, *radar*, and concepts directly related to these concepts: *Darth Vader*, *dark*, *dangerous*, *deep*, *evil*, *cat*, *airplane*, *sky*, *nothing*, ... Several of these have a short path [9] in the network to *dark*, and *dark* is directly related to *creepy*. The sum of the shortest path length to *creepy* is significantly less than (for example) the path score of the cluster defining *bunny*. A bat has many dark aspects, and dark is pretty creepy.

Note that different concepts in a cluster have a higher or lower influence on the final score. For the bat concept, the distance between *dark* and *creepy* is more essential than the distance between *deep* and *creepy*. This is because *dark* is more central in the bat cluster when we calculate its betweenness centrality [3]. More connections between concepts in the cluster pass through *dark*. We take *dark* as a sort of conceptual glue when reasoning about bats.

Using conceptual association, we think the system can be useful for human designers to come up with more creative ideas, or to find visual solutions for abstract concepts (e.g., *jazz* = *blue*).

5 Future Work

Section 4 presents a preliminary computational approach to simulate brainstorming. In future research, we will investigate if this is indeed how human brainstorming works, and if the analogies the system comes up with are "good" or "bad" creative finds. This will introduce new challenges, since what is "good" or what is "bad" appears to involve numerous cultural and personal factors.

Acknowledgements

The Gravital research project was funded by the agency for Innovation by Science and Technology (IWT), Belgium, 2007-2009. City In A Bottle is funded by the Flemish Audiovisual Fund (VAF), Belgium, 2008-2011.

References

1. Blackwell, T.M., Bentley, P.: Improvised music with swarms. In: Proceedings of the Evolutionary Computation on 2002. CEC 2002. Proceedings of the 2002 Congress, vol. 02, pp. 1462–1467 (2002)
2. Boden, M.: What is generative art? Digital Creativity 20, 21–46 (2009)
3. Brandes, U.: A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology (2001)
4. Chalmers, D.J., French, R.M., Hofstadter, D.R.: High-Level Perception, Representation, and Analogy: A Critique of Artificial Intelligence Methodology. Journal of Experimental & Theoretical Artificial Intelligence (1991)
5. Cleveland, P.: Bound to technology - the telltale signs in print. Design Studies 25(2), 113–153 (2004)
6. Collins, A.M., Loftus, E.F.: A Spreading-Activation Theory of Semantic Processing. Psychological Review 82(6), 407–428 (1975)
7. Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A. (2004). Timbl: Tilburg memory-based learner. In: Tech. Report ILK 04-02, Tilburg University (December 2004)
8. Daelemans, W., van den Bosch, A.: Memory-based language processing. In: Studies in Natural Language Processing, Cambridge University Press, Cambridge (2005)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
10. Gielis, J.: A generic geometric transformation that unifies a wide range of natural and abstract shapes. American Journal of Botany 90, 333–338 (2003)
11. Goldstein, J.: Emergence as a Construct: History and Issues. Emergence: Complexity and Organization 1, 49–72 (1999)
12. Hu, J., Goodman, E.: The hierarchical fair competition (HFC) model for parallel evolutionary algorithms. In: Proceedings of the Congress on Evolutionary Computation, pp. 49–54 (2002)
13. Kauffman, S.A.: The origins of order: self-organization and selection in evolution. Oxford University Press, Oxford (1993)
14. McCormack, J., Dorin, A.: Art, Emergence and the Computational Sublime. In: A Conference on Generative Systems in the Electronic Arts, CEMA, Melbourne, Australia, pp. 67–81 (2001)
15. Mednick, S.: The Associative Basis of the Creative Process. Psychological Review 69, 220–232 (1962)
16. Olivero, G.: Spamghetto. In: Data Flow 2, Visualizing Information in Graphic Design, p. 165. Gestalten, Berlin (2010)

17. Perlin, K.: Improving noise. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002), pp. 681–682 (2002)
18. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, Heidelberg (1990)
19. Reas, C., Fry, B.: Processing: a programming handbook for visual designers and artists. MIT Press, Cambridge (2007)
20. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics 21(4) (1987)
21. Schank, R.C., Abelson, R.P.: Scripts, plans, goals, and understanding: an inquiry into human knowledge structures. Lawrence Erlbaum Associates, Mahwah (1977)
22. Sowa, J.F.: Semantic Networks. In: Shapiro, S.C. (ed.) Encyclopedia of Artificial Intelligence. Wiley, New York (1987); revised and extended for the second edition (1992)
23. Winograd, T.: Understanding Natural Language. Academic Press, London (1972)

Evolving Four-Part Harmony Using Genetic Algorithms

Patrick Donnelly and John Sheppard

Department of Computer Science, Montana State University, Bozeman MT 59715
{patrick.donnelly2,john.sheppard}@cs.montana.edu

Abstract. This paper presents a genetic algorithm that evolves a four-part musical composition–melodically, harmonically, and rhythmically. Unlike similar attempts in the literature, our composition evolves from a single musical chord without human intervention or initial musical material. The mutation rules and fitness evaluation are based on common rules from music theory. The genetic operators and individual mutation rules are selected from probability distributions that evolve alongside the musical material.

Keywords: Genetic Algorithm, Evolutionary Programming, Melody, Harmony, Rhythm, Music Composition.

1 Introduction

Algorithmic composition is the process of creating music using either determinate or indeterminate algorithmic processes, or some combination thereof. While composing, musical composers use many simple rules during the creative process. Unlike stochastic composition techniques, algorithmic methods that encode knowledge from rules of voice leading and counterpoint can better approximate musically desirable solutions.

The research presented here attempts to compose four-part musical harmony with a GA without user intervention or initial musical material. Each musical piece begins as a single musical chord and is expanded using genetic operations. Unlike previous music experiments with GAs that focus on one aspect of music or require initial musical material, such as a melody to harmonize, our system evolves the entire composition, including the melodies, harmonies, and rhythms, using genetic operations. Our system features a variable length chromosome that represents a four-voice composition and uses the four basic genetic operators of mutation, duplication, inversion, and crossover. The mutation operator and the fitness function evaluation are based on music theory rules.

2 Related Work

Algorithmic composition has long been of interest to computer scientists and musicians alike. Researchers have used a variety of approaches toward algorithmic

© Springer-Verlag Berlin Heidelberg 2011

composition including rule-based systems [2], musical grammars, artificial neural networks, and Markov chains (see [11] for a review).

Previously, genetic algorithms have been applied in several areas of music composition. However, given the broad search space afforded by GAs and the multi-dimensional complexity of music itself, most studies have constrained themselves to focus on one dimension of music at a time, such as evolving rhythms, melodies, or harmonies, but we have found no studies that evolve all aspects together. Genetic algorithms have been used to compose single line melodies [3], creative jazz melodies based on a given chord progression [1], unpitched rhythms [5], Baroque-style harmony based on a given melody [10] [12] and figured bass [8], and the composition of microtonal music [6]. Compositional rules based on music theory have been used to control the fitness function [9] as well as the mutation operator [13] in the evolution of single line melodies.

As discussed in [11], the efficacy of any algorithmic composition method depends heavily on the underlying representation of musical knowledge. Most approaches to composition using GAs apply mutation and other operators stochastically, relying on selection and the fitness score to eliminate the weak individuals from the population. Our approach uses an probabilistic approach to operator selection, but unlike previous approaches, we allow these probabilities to evolve alongside the musical material. Therefore the individuals that succeed to subsequent generations contain information about which musical rules and genetic operators contributed to their higher fitness scores, thus allowing a more directed and efficient traversal of the broad evolutionary search space.

3 Genetic Algorithm

3.1 Genome Representation

Individual in the population consist of a variable length chromosome that represents a single musical composition. A chromosome contains four parts, each corresponding to a musical line. Every individual part consists of a list of tuples containing a pitch and duration value. Musical pitch is represented by a standard MIDI value, ranging $[0, 127]$. Duration is represented as an integer from $[1, 8]$ that corresponds to the length in semiquavers (eighth notes). A duration value of one indicates a single eighth note while a duration of eight represents a single whole note (Fig. 1). The musical representation contains no time signature, allowing the composition to grow and shrink from insertions and deletions anywhere in the piece. The chromosome also assumes a key of C-Major and does not contain any key modulations. The final output can be transposed to another musical key.

Each individual in the initial population consists of only a single whole note chord. The bass line always contains the tonic note (C) while each of the upper three parts contains a single note in the tonic chord selected randomly with replacement from the C-Major triad (C, E, or G). Each musical part will grow from this single initial note using mutation and the other genetic operators.

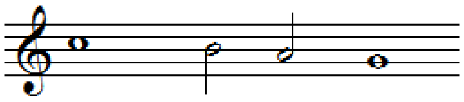


Fig. 1. Example chromosome for a single part represented as a list of pitch-duration tuples: $\{(72, 8); (71, 4); (69, 4); (67, 8)\}$

3.2 Operators

Operator Probability. In addition to the musical information, each chromosome also contains a set of probabilities used in the selection of the genetic operators. Each genetic operator *op* has an individual probability drawn from a total probability mass.

$$\mathbf{P}(op) = \begin{cases} P(mutation) \\ P(crossover) \\ P(duplication) \\ P(inversion) \end{cases} \in [0,1]$$

$\mathbf{P}(op)$ represents the distribution that operator *op* will be selected. On each generation and for each chromosome, a single operator is probabilistically selected and applied. The probabilities of the genetic operators are themselves mutated in every generation. A probability corresponding to a single operator is randomly selected from the distribution, and a random value from $[-0.1, 0.1]$ is added to the probability. The distribution is then normalized to maintain a sum of one. The initial operator probability distribution is shown in Table 1.

Like the operator probabilities, individual mutation rules are also selected probabilistically. On each generation, if the mutation operator is selected, a single mutation rule m_i is selected probabilistically and applied to the chromosome. These individual mutation rules are drawn from a separate probability distribution, where

$$\sum_i P(m_i | op = mutation) = 1$$

Thus, the probability of selecting a specific mutation operator is given by $P(m_i, mutation) = P(m_i | mutation) \times P(mutation)$. The probabilities of the mutation rules are randomly perturbed each generation in the same manner as the operator probabilities. The probability distributions for the mutation rule selection are initialized uniformly with an equal fraction of the total distribution.

Selection Method. To create a new generation, the fitness score of each chromosome is calculated. Truncation elimination is applied to remove the weakest ten percent from the population. To preserve the most fit chromosomes, an elitist selection adds the top ten percent of the population to the successive generation without any change.

Table 1. Initial Operator Probabilities

Operator	Initial Probability
Mutation	0.4
Crossover	0.2
Inversion	0.2
Duplication	0.2

The remainder of the population is generated by altering chromosomes using one of the four genetic operators. A single chromosome from the previous generation is selected by tournament selection with a tournament size of two. This selected chromosome is then altered by one of the genetic operators probabilistically and added to the subsequent generation.

Mutation. If the mutation operator is selected, a single mutation rule is selected according to the rules probability. For each part, a note or pair of consecutive notes is selected at random and the mutation rule is applied. The mutation rules are described below.

1. **Repeat** – the selected note is repeated with the same pitch and duration value.
2. **Split** – the selected note is split into two notes. Each new note has the same pitch as the original, but half the original duration.
3. **Arpeggiate** – the selected note is split into two notes of equal duration. The first note retains the original pitch value while the second note is transposed randomly to pitch a third or fifth above the first note.
4. **Leap** – the pitch of the selected note is randomly changed to another pitch in the C-Major scale within the musical range of the part.
5. **Upper Neighbor** – for a pair of consecutive notes with the same pitch, the pitch of the second note is transposed one diatonic scale step higher than the original.
6. **Lower Neighbor** – for a pair of consecutive notes with the same pitch, the pitch of the second note is transposed one diatonic scale step lower than the original.
7. **Anticipation** – the selected note is split into two notes, each containing the original pitch, but an unequal duration. The duration of the first note is shorter than the second note by the ratio of 1:3. For example, if the original note is a half note, the new notes will be an eighth and a dotted-quarter note.
8. **Delay** – the selected note is split into two notes, each containing the original pitch, but of unequal duration. The duration of the first note is longer than the second note by the ratio of 3:1.
9. **Passing Tone(s)** – for a pair of selected consecutive notes, new notes are added in between that connect the two notes by stepwise motion. If the second note is lower than the first, the new notes connect in downward scalar motion. If the second is higher than and the first, the new notes connect the two original notes by upward scalar motion.

10. **Delete Note** – the selected note is removed from the part.
11. **Merge Notes** – for a pair of consecutive notes with the same pitch, the two notes are combined into a single note with a duration that is the sum of the two original notes, or eight, whichever is smaller.

Crossover. When the crossover operator is chosen, a pair of chromosomes is selected by tournament selection and recombined using single-point crossover to create two new chromosomes. For any selected pair of chromosomes C^1 and C^2 with lengths m and n respectively, cutpoints i and j are selected at random. A cutpoint is a duration point selected randomly anywhere between the start and the end of a piece. It may fall on a boundary between two notes, or bisect an individual note, in which case the note is split into two notes at this boundary. Two new chromosomes $C_3 = C_1^1..C_i^1C_{j+1}^2..C_n^2$ and $C_4 = C_1^2..C_j^2C_{i+1}^1..C_m^1$ are created and added to the next generation (Fig. 2(a)). The crossover operator is applied to each of the parts using the same cutpoints. The operator probabilities and the mutation rule probabilities for the new chromosome are the average of the respective probabilities of the original two chromosomes.

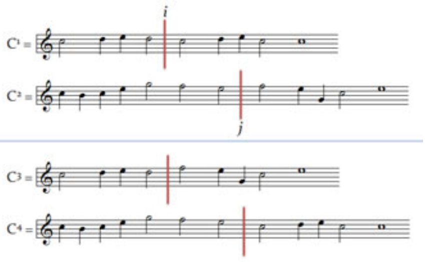
Duplication. A single chromosome is selected by tournament selection and a portion of the chromosome is repeated. An initial cutpoint i is selected at random and a second cutpoint j is selected at random, such that $j > i$ and $j < i + 8$. The duplication operator restricts repetition to at most the duration of a whole note so that the chromosome does not grow in length disproportionately with other individuals in the population. A new chromosome containing a repetition of notes from i to j is created and added to the next generation (Fig. 2(b)).

Inversion. A single chromosome C is selected by tournament selection and a subsequence of the chromosome is reversed, similar to John Holland's original inversion operator [4]. An initial cutpoint i is selected at random and a second cutpoint j is selected at random, such that $j > i$. The order of the notes between i and j are then reversed. The new chromosome containing the retrograde musical subsequence is added to the next generation (Fig. 2(c)).

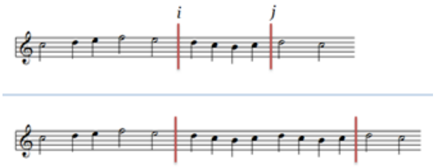
3.3 Fitness Function

The fitness function evaluates the quality of various aspects of the piece of music according to several common rules of musical composition and voice leading [7]. As in [9], the fitness of the entire chromosome is a weighted sum of R individual fitness rules. In order to balance the relative importance of some rules over others, each rule has a weight associated with it. Each rule is analyzed separately. An individual fitness rule r_i produces a $score(r_i) \in [0, 1]$, and w_i indicates its corresponding weight in the total fitness score.

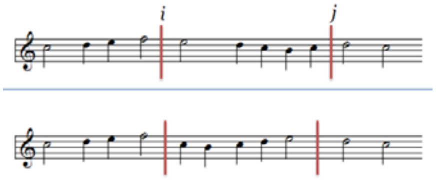
$$fitness = \sum_{i}^{R} w_i \times score(r_i)$$



(a) Example of the crossover operation. Above, original chromosomes C^1 and C^2 with cutpoints i and j, respectively. Below, new chromosomes C^3 and C^4 resulting from crossover of C^1 and C^2.



(b)) Example of the duplication operation. Above, the original chromosome with cutpoints i and j. Below, the new chromosome with duplicated subsequence.



(c) Example of the inversion operator. Above, the original chromosome with cutpoints i and j. Below, the new chromosome with the inverted subsequence.

Fig. 2. Example of the crossover, duplication, and inversion operators for a single musical line

An individual fitness rule indicates how well the composition conforms to the individual rule alone. We analyze each opportunity n that the rule might be analyzed and each violation v of the rule is counted.

$$score(r_i) = \frac{n - v}{n}$$

For rules that involve only a single note, such as Part Range, n corresponds to a count of the notes in the composition. However, for rules that must analyze two notes, such as Leap Height, n corresponds to the sum of the number notes in each part minus one. A score of one indicates there were no rule violations while a value of zero indicates the rule was violated at every opportunity.

In our present experiments, we naïvely weight the rules according to the type of rule: melody, harmony, or rhythm. The sum of the rules within a type are normalized to value between $[0, 1]$, and the total fitness score of the chromosome is the sum of these three types, a value between $[0, 3]$. In future work, we wish to experimentally tune the weights as in [9] by empirically testing each fitness rule individually on compositions from the literature and weighting each rule according to its potential for fitness score gains.

Melodic Rules. The following rules examine pitch within a single part:

1. **Opening Chord** – the first chord should be a tonic chord.

2. **Closing Chord** – the final chord should be a tonic chord.
3. **Final Cadence** – the last two chords should form an authentic cadence in which the bass line should close with movement from a fifth above or a fourth below to the tonic note and the other three parts should close by stepwise motion to the final note.
4. **Part Range** – the difference between the lowest note and the highest note in each part should be no more than an interval of a 13th (octave and a fifth).
5. **Leap Height** – individual leaps should be no more than an interval of a 9th (octave and a second).
6. **Leap Resolution** – leaps more than an interval of a major sixth should resolve in the opposite direction by stepwise motion.
7. **Voice Crossing** – a lower part should not contain a pitch higher than an upper part and an upper part should not contain pitches below a lower part.
8. **Voice Range** – each part should not contain pitches outside its range (Fig. 3).
9. **Repeating Pitch** – no part should repeat the same pitch more than three times consecutively.
10. **Stepwise Motion** – at least half of all melodic intervals in each part should be stepwise motion. This helps forms a coherent melodic line by penalizing excessive leapiness.

Harmonic Rules. The following rules examine pitch across multiple parts:

1. **Parallel Octaves** – no two parts should contain motion that forms parallel octaves (two parts one octave apart moving to new notes also an octave apart).
2. **Parallel Fifths** – no two parts should contain motion that forms parallel fifths.
3. **Vertical Consonance** – no two parts should form a dissonant harmonic interval (seconds, fourths, or sevenths).

Rhythm Rules. The following rules examine duration within a single part:

1. **Rhythmic Variation** – the piece should contain instances of each note duration type, from the eighth note to the whole note.
2. **Note Repetition** – the piece should not repeat a note of the same duration more than three times consecutively.

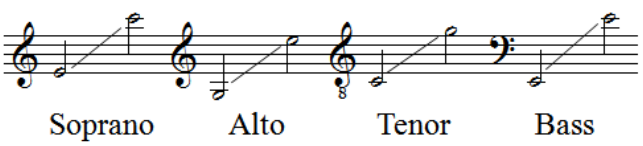


Fig. 3. Musical range of each part

4 Results

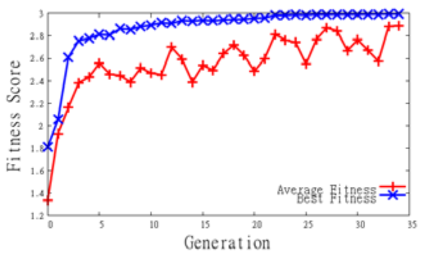
In our experiments, we use a population size of 1000. On each generation we calculate the average fitness of the entire population. The genetic algorithm is halted when the average fitness of the entire population converges to a value no more than 0.01 from the previous generations average fitness. The chromosome with the highest fitness score is saved as a MIDI file. In an experimental run of 100 trials, the average number of generations was 11.06 and the average fitness score of the final generation was 2.668 out of a maximal possible fitness score of 3.0. Fig. 4(a) shows the average and best fitness scores for a single experimental run of 34 generations.

We examined the evolution of the individual operator probabilities throughout the evolutionary process. In each generation, we average each of the four operator probabilities over the entire population. At the final generation, the probability of the crossover operator dominated the distribution, accounting for approximately 97% of the probability mass, leaving only trivial probabilities of the other three operators. In a separate experiment, we examined adding a fifth operator that consisted of no operation, allowing an individual to move to the next generation unchanged. However, even with this additional operator the crossover probability continued to dominate. This annealed preference of the crossover operator demonstrates that crossover presents the best opportunity for fitness score gains (Fig. 4(b)).

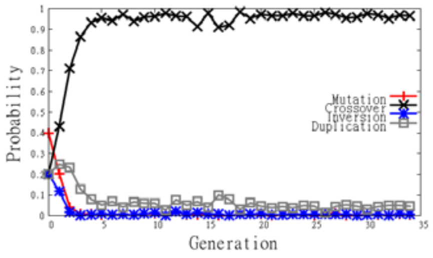
Our preliminary results demonstrate that it possible to evolve four-part musical compositions entirely from an initial single chord (Fig. 5). The resulting compositions show a number of desirable musical qualities, such as preference for leaps in the outer parts, preference for stepwise motion in the inner parts, contrary motion between parts, and singable parts in the proper musical ranges. While our compositions are musical, they suffer from several limitations. Since we allow each musical line to grow or shrink independently, analyzing the harmony only in the fitness evaluation, the parts tend to move independently and the composition often lacks coherence between parts. Additionally, our results feature a dominance of C-Major chords, although Fig. 5 indicates there are some exceptions. Although the mutation rules do change individual pitches, the resulting harmony rule score tends to penalize the composition before pitches in the other parts have a chance to mutate and form a new chord. These limitations can be overcome with a better tuning of the fitness score weights and the addition of new mutation rules.

5 Conclusion and Future Work

In this paper we present a method to evolve four-part musical compositions using genetic algorithms. While there have been other attempts in the literature to compose music using GAs, these works generally constrain themselves to one aspect of music alone. Our work examined the feasibility of using an evolutionary technique to create entire music compositions, including the melodic lines,



(a) Average fitness score and the best fitness score for each generation.



(b) Operator probabilities averaged over entire population.

Fig. 4. Experimental run of 34 generations

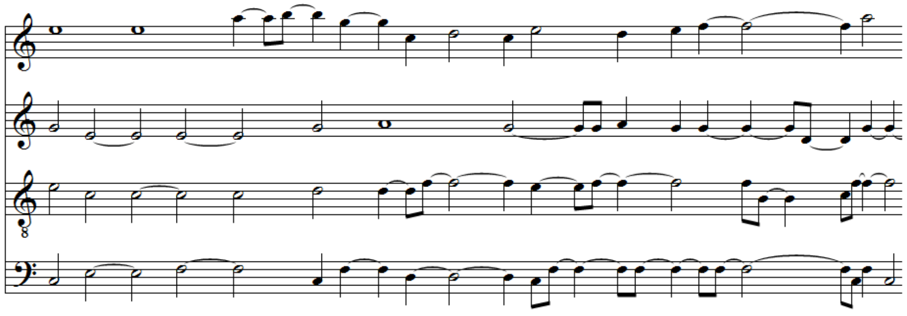


Fig. 5. Excerpt of a composition with a fitness score of 2.94

rhythms, and harmonies, from a single musical chord, using mutation rules and fitness evaluation based on rules of music composition. Another major difference is that our work evolves the probabilities of the operator and mutation rule selection. As the most fit individuals of the population survive to reproduce from generation to generation, their probabilities most often reflect the rules which contributed to their high fitness and are probabilistically more likely to be used again.

As future work, we plan to examine and encode more mutation and fitness rules based on more complicated rules from music theory, such as examining melodic contour, encouraging contrary motion of the parts, as well as a more complicated analysis of the chords in the resulting harmony. Furthermore, we also plan to encode a key-signature in the chromosome to allow for the evolution of richer harmonies and more complicated chord progressions. We will also examine using our system to evolve individual musical phrases, embedded within a second higher-level GA that will combine individual phrases into a longer composition including key-modulations and well-defined cadences. Lastly, to improve our fitness function and balance the many fitness rules we employ, we will empirically test the fitness rules on a selection of compositions from the Renaissance and Baroque periods to experimentally determine a set of weights.

References

1. Biles, J.: GenJam: A genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference, pp. 131–131 (1994)
2. Cope, D.: Virtual Music: Computer Synthesis of Musical Style. The MIT Press, Cambridge (2004)
3. Göksu, H., Pigg, P., Dixit, V.: Music composition using Genetic Algorithms (GA) and multilayer perceptrons (MLP). In: Advances in Natural Computation, pp. 1242–1250 (2005)
4. Holland, J.: Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor (1975)
5. Horowitz, D.: Generating rhythms with genetic algorithms. In: Proceedings Of The National Conference On Artificial Intelligence, pp. 1459–1459. John Wiley & Sons, Chichester (1995)
6. Jacob, B.: Composing with genetic algorithms, pp. 452–455 (1995)
7. Kostka, S., Payne, D., Schindler, A.: Tonal harmony, with an introduction to twentieth-century music. McGraw-Hill, New York (2000)
8. Maddox, T., Otten, J.: Using an Evolutionary Algorithm to Generate Four-Part 18th Century Harmony. Mathematics and Computers in Modern Science: Acoustics and Music, Biology and Chemistry, Business and Economics, 83–89 (2000)
9. Marques, M., Oliveira, V., Vieira, S., Rosa, A.: Music composition using genetic evolutionary algorithms. In: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, pp. 714–719 (2000)
10. McIntyre, R.: Bach in a box: The evolution of four part baroque harmony using the genetic algorithm. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 852–857 (2002)
11. Papadopoulos, G., Wiggins, G.: AI methods for algorithmic composition: A survey, a critical view and future prospects. In: AISB Symposium on Musical Creativity, pp. 110–117 (1999)
12. Phon-Amnuaisuk, S., Wiggins, G.: The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In: Proceedings of the AISB 1999 Symposium on Musical Creativity (1999)
13. Towsey, M., Brown, A., Wright, S., Diederich, J.: Towards melodic extension using genetic algorithms, pp. 54–64. International Forum of Educational Technology & Society (2001)

A Sonic Eco-System of Self-Organising Musical Agents

Arne Eigenfeldt[1] and Philippe Pasquier[2]

[1] School for the Contemporary Arts, Simon Fraser University
Vancouver, Canada
[2] School for Interactive Arts and Technology, Simon Fraser University,
Surrey, Canada
{arne_e,pasquier)@sfu.ca

Abstract. We present a population of autonomous agents that exist within a sonic eco-system derived from real-time analysis of live audio. In this system, entitled Coming Together: Shoals, agents search for food consisting of CataRT unit analyses, which, when found, are consumed through granulation. Individual agents are initialised with random synthesis parameters, but communicate these parameters to agents in local neighborhoods. Agents form social networks, and converge their parameters within these networks, thereby creating unified grain streams. Separate gestures thus emerge through the self-organisation of the population.

Keywords: Sonic eco-system, Artificial-Life, self-organisation.

1 Introduction

Artificial-Life (A-Life), specifically the properties of self-organisation and emergence often found within it, offers composers new paradigms for computer music composition. The temporal nature of emergence — one desirable outcome of A-Life systems — provides a parallel to the complexity and evolution of gestural interactions sought by composers of both fixed and non-fixed music. Composers of generative computer music have found A-Life to be a particularly fruitful area of investigation. As McCormack [12] proposes, a successful evolutionary music system can "enable the creative exploration of generative computational phase-spaces."

1.1 Eco-Systems versus Evolution

Martins and Miranda point out that, while A-Life offers new possibilities for computer music composition, its algorithms must be adapted to suit its musical ends. Miranda's four requirements for evolution [15] are actually extremely difficult to achieve, particularly in a real-time context: his first criteria — selection of transformations — requires a system that can overcome the fitness bottleneck of interactive evaluation [2], a task that is difficult, if not impossible, given the aesthetic problem of evaluating successful musical evolution.

Bown [4] discusses some of the failures of the traditional A-Life paradigm of the interactive genetic algorithm, and points to new approaches based within social learning, cultural dynamics, and niche construction that offer potential solutions [13].

© Springer-Verlag Berlin Heidelberg 2011

Bown suggests that an eco-system approach "might generate sonic works that continue to develop and transform indefinitely, but with consistent structural and aesthetic properties". McCormack and Bown [13] posit that an eco-system approach emphasizes the design of interaction between its components, and that it is conceptualized within its medium itself: a sonic eco-system operates in the medium of sound rather than being a sonification of a process. RiverWave [13] is an example installation that demonstrates their concept of a sonic eco-system.

We have similarly chosen to avoid mating, reproduction, and selection within our system — all traditional properties of A-Life and evolutionary computing — instead focusing upon the complex interactions of an existing population within a sonic eco-system. Like Bown, we are interested in musical evolution and self-organisation of musical agents in real-time, in which the dynamic evolution is the composition [9]. Section 2 will discuss related work, including the use of musical agents, A-Life models for sound synthesis, and the uniqueness of our research; Section 3 will present a detailed description of the system; Section 4 will offer a conclusion and future research.

2 Related Work

We build upon the research into Artificial-Life based in audio by Jon McCormack [11], Joseph Nechvatal [19], Eduardo Miranda [15], Peter Beyls [1], Oliver Bown [4], and Tim Blackwell and Michael Young [3].

2.1 Musical Agents

The promise of agent-based composition in musical real-time interactive systems has been suggested [23, 18, 15], specifically in their potential for emulating human performer interaction. The authors' own research into multi-agent rhythm systems [6, 7, 8] has generated several successful performance systems. Agents have been defined as autonomous, social, reactive, and proactive [22], similar attributes required of performers in improvisation ensembles.

Martins and Miranda [10] describe an A-Life system in which users can explore rhythms developed in a collective performance environment. This system is an evolution of earlier work [16] in which agents could be physical robots or virtual entities whose data consisted of sung melodies. In this later research, agents are identical and remain virtual, although the number of agents can vary. Agents move in 2D space, but their interaction is limited to pairs exchanging data. Although the data is limited to rhythmic representations, the resulting transformations suggest a convergence similar to the system described in this paper; however, Martins and Miranda's motivation is extra-musical: "these transformations were inspired by the dynamical systems approach to study human bimanual coordination and is based on the notion that two coupled oscillators will converge to stability points at frequencies related by integer ratios." Agents build a repertoire of rhythms that will eventually represent a population's preference; however, these rhythms are not played collectively at any time, unlike the system described in this paper, in which a population's current state is immediately audible.

Martin and Miranda's conclusion points out problems with current approaches to musical composition with A-Life: "the act of composing music seldom involves an automated selective procedure towards an ideal outcome based on a set of definite fitness criteria." In this case, the creation of rhythms may have followed a evolutionary path that utilised complex social interactions, but the musical application of this data was a simple random selection. While the system demonstrates that "social pressure steers the development of musical conventions", the pressure is unmusical: it aims toward homogeneity.

Bown describes a system [5] which "animates the sounds of a man-made environment, establishing a digital wilderness into which these sounds are 'released' and allowed to evolve interdependent relationships". The analysed acoustic environment of the installation space, which includes the projected sounds of the agents, is treated as the virtual environment in which the agents exist. A continuous sonogram is made of the acoustic space, and agents attempt to inhibit low-energy regions, thereby creating an interesting dilemma: as the agents make sound at their occupied frequency region, they deplete the available resources at that region. It is the interaction of agents competing for available resources that creates the potential for emergence, and thus provides the musical evolution.

2.2 A-Life Models for Sound Synthesis

The system described in this paper utilizes granular synthesis as its method of sound generation, with agent interactions within the eco-system determining the synthesis parameters. Miranda's ChaosSynth [14] was one of the first systems to use models other than stochastic processes for parameter control of granular synthesis: in this case, cellular automata.

Blackwell and Young [3] have investigated the potential for swarms as models of for composition and synthesis, pointing to the similarity between the self-organisation of swarms, flocks, herds, and shoals and that of improvising musicians. The authors suggest that improvising music can explore musical aspects often ignored in accepted musical representations of "the note" — namely timbre and its morphology — aspects often shared by composers working with computers. The authors extend swarm organisation for synthesis in their Swarm Granulator, which, like ChaosSynth and the system described here, uses granular synthesis.

3 Description

3.1 Audio Analysis

We use a modified version of CataRT [21] as a real-time analysis method to generate segmentation and audio descriptors of live audio. Equally segmented units are stored in an FTM data structure [20] that has been extended to include a 24-band Bark analysis [24]. CataRT plots a 2-dimensional projection of the units in the descriptor space: we have chosen spectral flatness and spectral centroid (see Figure 1).

Incoming audio is initially recorded into a gated buffer in MaxMSP — silences and low amplitudes are not recorded — which is passed to CataRT every 12.1 seconds, resulting in 50 units of 242 ms in duration per analysis pass. When analysis is complete (at faster than real-time speed), the new units immediately populate the space. CataRT tags all units within an analysis pass with an index, or `SoundFile` number, which can be used to determine the most recent units.

We also use CataRT to query agent proximity to units, a function that returns a list of units within a defined radius (see Section 3.2.2).

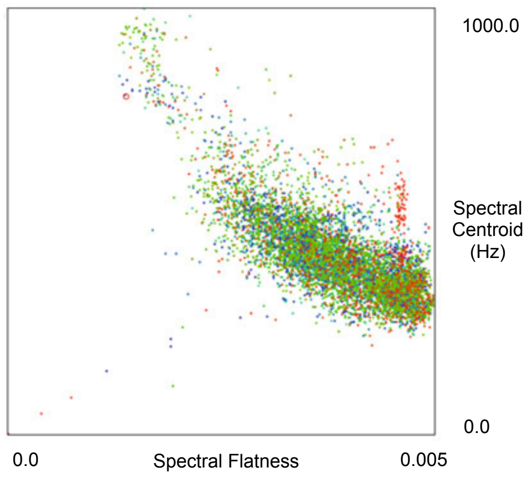


Fig. 1. CataRT display of units in 2D space, using spectral flatness (x) and spectral centroid (y)

3.2 Agents

A variable number of agents[1] exist within the space, moving freely within it, interacting with one another, and treating the CataRT units as food. When agents find food, they consume it by using its audio as source material for granular synthesis. How the agents move within the space, and how they play the audio once found, is dependent upon internal parameters that are set randomly during initialisation. As agents interact with one another, they share their parametric data, allowing for convergence over time (see Section 3.3). As the agents themselves converge in different areas within the space, their eventual parametric similarities, coupled with related spectra due to their locations, produces differentiated gestures.

3.2.1 Initialisation

When the system is initialised, agents select a random location within the space, a location which is immediately broadcast to the entire population. Agents store other agent locations, and calculate proximity to other agents after each movement step, or when finished playing a phrase. Those agents within a globally defined radius (`gNRadius`) of one another are considered to be within a neighborhood.

At initialisation, agents also select random values for their synthesis parameter ranges, from the following limits:

1. Grain duration (20-250 ms);
2. Delay between grains (5-350 ms);
3. Amplitude (0. - 1.);
4. Offset into the sample (0 to 1., where 1. is the duration of the sample less the current grain duration);
5. Phrase length (4 - 100 grains);
6. Pause between phrases (0 - 2500 ms);

[1] We have run as high as 48 agents on a single computer without obvious latency.

7. Phrase type (how subsequent grain delays are calculated within a phrase: stochastically using white noise, stochastically using brown noise, or exponential curves);
8. Output (number of speakers: for example, 1-8).

An agent broadcasts its parameters to the entire population if it is actively playing.

Lastly, agents select random values for their existence within the environment, including:

1. Acquiescence (the desire to stay with the same food, given a variety of nearby food sources);
2. Sociability (the desire to form social networks).

3.2.2 Movement

Agents are considered active if they have found food — the CataRT units — and are generating sound (see Section 3.2.4); only inactive agents move. Agents move at independent clock rates, calculating a time within their current pause-between-phrases range (initially two values randomly chosen between 0 and 2500 ms) from which a random selection is made. Agents move one grid location within the toroidal grid, then select a new delay time for their next movement phase.

Agents can see food outside their current grid location at a distance dependent upon a global radius: `gRadius`. When an agent sees food, it moves toward it. If no food is within view, the agent randomly selects a new location from its eight neighboring grid spaces, a selection negatively weighted by any locations the agent previously occupied. Agents keep track of their previous locations: if they have occupied a grid space in which no food was found, that location's desirability is decremented in the agent's location history. When selecting a new grid space, previously explored spaces have a lower likelihood of being selected. This allows for more efficient exploration of local space, with minimal backtracking (see Figure 2).

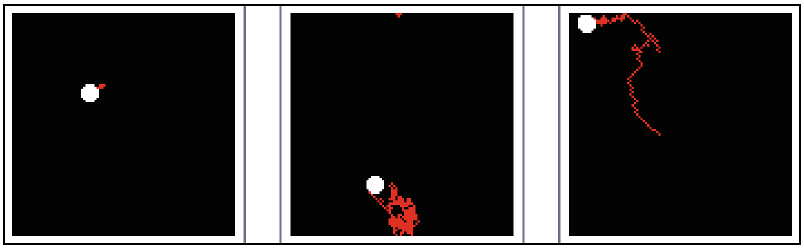


Fig. 2. Three agent's movements over time. The agent in the centre has found food after methodically exploring nearby space; the agent to the right is still looking for food; the agent to the left has recently been reincarnated, and thus has a short movement history.

When audio is recorded, the environment briefly becomes 3 dimensional, with the third dimension being time: the most recent `SoundFiles` appear at the top of the 3D cube. New audio analyses trigger an "excitement" mode, which allows the agents to move at much faster rates (using their grain delay times, rather than phrase delay) in search of new food. Any agent currently active will become inactive after its phrase is complete, and also move toward the new food. An analogy can be made of a fish-tank, with new food appearing at the surface of the tank, and the fish inside the tank swimming upwards toward it. The excited mode lasts an independent amount of time for each agent (exponentially weighted around a global variable `gExcitementTime`) after which the agents return to normal speed. The purpose of the excited mode is twofold: to allow non-active wandering agents to find food faster, and to force active agents to search for more recent audio. In terms of CataRT, during the excited mode, the most recent `SoundFile` number is included in the query.

When an agent finds food, it broadcasts this information to its neighbors. All inactive agents within that individual's neighborhood can then move toward the location in which food was discovered (see Figure 3). However, if an agent does not find food within a defined number of movement turns (a global variable that translates into a global strength, or constitution, of all agents in the population), the agent dies. When this happens, the agent broadcasts its impending death to the population, which remove that agent's data from their internal arrays. After a length of time (a global variable range between 5 and 60 seconds), the agent is reincarnated at a new location with new parameter data. The decision to use a model of reincarnation, as opposed to one of evolving generations, was made since the latter would offer little to this model – it is the complex interactions of the community over time that are of significance.[2]

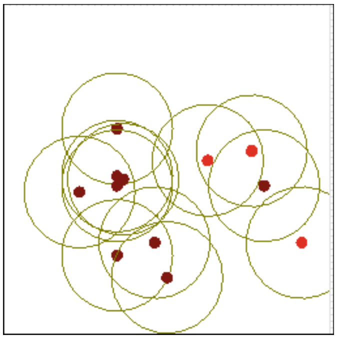


Fig. 3. Twelve agents within the space, with their individual neighborhood ranges visible. Dark agents are inactive, lighter agents are active. Inactive agents can move towards active agents within their neighborhood.

3.2.4 Sound Generation

Once an agent has moved with the CataRT's `gRadius` of a food source, the agent has access to both the unit's sample and analysis data. Using a custom granular synthesis patch created in MaxMSP, agents play a monophonic grain stream whose individual grain's amplitude, duration, delay between grains, and offset into the sample are determined by the agent's synthesis parameters. It was found that stochastic delays between grains did not significantly differentiate agent streams; therefore, over the course of an agent's phrase, the delays are chosen from the available delay range in one of three ways: white (evenly distributed throughout the

[2] On a more practical level, the number of agents cannot be dynamic within MaxMSP, as a set number need to be initialized before calculating the DSP chain.

range); brown (a random walk through the range); and curved (an exponential curve between range extremes). The specific probability distributions over the possible types are a converging parameter within the neighborhood (see Section 3.3.1).

The number of grains in a phrase is chosen from the agent's phrase range, while the length of time between grain phrases is chosen from the agent's pause range. A single agent's audio is rather meek, in that its amplitude is scaled by the number of agents within the eco-system. Interestingly, the stream itself somewhat resembles an insect-like sound due to its phrase length, discrete events, and monophonic nature[3].

The agent's spectral bandwidth is limited, as agents play their audio through a 24-band resonate filter, whose frequencies are set to those of the Bark analysis. The width of the filter is dependent upon the size of the agent's social network (see Section 3.3): agents outside a social network play the entire bandwidth, while agents within a network divide the bandwidth between them, selecting centre frequencies through negotiation (see Section 3.3). As the spectral bands of a unit are played, its Bark energy is lowered (at a rate dependent upon a globally set variable `gPersistence`); if a unit's bands are dissipated completely, the unit is removed from the environment. Agent's within small social networks therefore "use up" food faster, as their spectral range is wider; as such, an agent's fitness is dependent upon its ability to coexist with other agents.

3.3 Agent Interaction: Social Networks

Agents within a global radius of one another are considered to be within a neighborhood. Agents can be in multiple neighborhoods (see Figure 4).

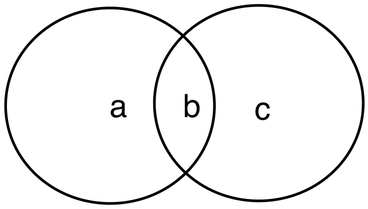


Fig. 4. Two neighborhoods, around agents a and c. Agent b is a member of both neighborhoods.

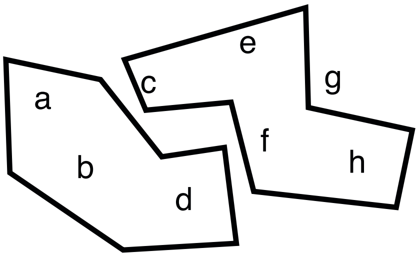


Fig. 5. Two social networks (a b d) and (c e f h). Agent g is not in a network.

[3] An example can be heard at `www.sfu.ca/~eigenfel/research.html`

After each movement cycle, or end of an audio phrase, agents can make a “friend-request” to another active agent in its neighborhood, in an effort to create a social network. The likelihood of this request being made is dependent upon the agent’s sociability rating, while the target is dependent upon that agent’s current social network status. If the agent has no friends in its own network, it will look for existing networks to join, favouring agents within larger networks. If an agent is already in a social network, it will look for agents outside of a network in an effort to make its own social network larger. Agents can choose to leave their network and join another, if the requesting agent’s network is larger than the agent’s own network.

Once agents have found food, and thus stopped moving, neighborhoods become static; however, social networks will continue to be dynamic, as agents can tempt each other away from existing networks (see Figure 5).

3.3.1 Convergence

Agents communicate their synthesis parameters within their social networks, and converge these parameters upon local medians. When an agent completes a grain stream phrase, it broadcasts which unit it consumed, along with the specific Bark bands. Next, it calculates the mean for all individual synthesis parameters it has accumulated from its social network, comparing it to its own parameters, and sets new values from the mean of the agent’s previous values and the group mean.

Once values have converged for a parameter to a point that an agent’s range is within 10% of the group mean range, for a period of time dependent upon a globally set variable `gDivergenceDelay`, divergence can occur, in which an agent can choose to select a new, random range. This alternation between convergence and divergence was used previously [9] in a multi-agent system to create continually varying, but related, group dynamics, and is an example of a heuristic decision that creates successful musical results. It can, however, be justified through biological models: agents can decide to leave a group if it becomes too crowded.

4 Conclusions and Future Work

Many different social networks emerge within a continuously running performance, aurally defined by their unified gestures due to their shared synthesis parameters and spectral properties. The musical goal of the eco-system is to move from independent individual granular streams into cohesive gestures, which depend upon similar spectra (arrived at through localization within the space) and similar synthesis parameters (arrived at through convergence).

Future work includes creating eco-systems across a high-speed network, in which agents act upon separate analysis created through different live audio. Related work includes sending audio generated by the eco-system across a high-speed network to an independent eco-system, complementing or replacing that eco-system’s live audio.

Acknowledgments

This research was funded by a grant from the Canada Council for the Arts, and the National Sciences and Engineering Research Council of Canada.

References

1. Beyls, P.: Interaction and Self-Organisation in a Society of Musical Agents. In: Proceedings of ECAL 2007 Workshop on Music and Artificial Life, Lisbon (2007)
2. Biles, J.: Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness. In: Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program, San Francisco (2001)
3. Blackwell, T., Young, M.: Swarm Granulator. 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. 399–408. Springer, Heidelberg (2004)
4. Bown, O.: A Framework for Eco-System-Based Generative Music. In: Proceedings of the SMC 2009, Porto, pp. 195–200 (2009)
5. Bown, O.: Eco-System Models for Real-time Generative Music: A Methodology and Framework. In: Proceedings of the ICMC 2009, Montreal, pp. 537–540 (2009)
6. Eigenfeldt, A.: Emergent Rhythms through Multi-agency in Max/MSP. In: Computer Music Modeling and Retrieval: Sense of Sounds, CMMR, pp. 368–379 (2008)
7. Eigenfeldt, A., Pasquier, P.: A Realtime Generative Music System using Autonomous Melody, Harmony, and Rhythm Agents. In: 12th Generative Art Conference Milan (2009)
8. Eigenfeldt, A.: The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 498–507. Springer, Heidelberg (2009)
9. Eigenfeldt, A.: Coming Together - Composition by Negotiation. In: Proceedings of ACM Multimedia, Firenze (2010)
10. Martins, J., Miranda, E.R.: Emergent rhythmic phrases in an A-Life environment. In: Proceedings of ECAL 2007 Workshop on Music and Artificial Life, Lisbon (2007)
11. McCormack, J.: Eden: An Evolutionary Sonic Ecosystem. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, pp. 133–142. Springer, Heidelberg (2001)
12. 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)
13. McCormack, J., Bown, O.: Life's what you make: Niche construction and evolutionary art. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 528–537. Springer, Heidelberg (2009)
14. Miranda, E.R.: Granular synthesis of sounds by means of cellular automata. Leonardo 28(4), 297–300 (1995)
15. Miranda, E.R.: Evolutionary music: breaking new ground. In: Composing Music with Computers. Focal Press (2001)
16. Miranda, E.R.: At the Crossroads of Evolutionary Computation and Music. Evolutionary Computation 12(2), 137–158 (2004)
17. Miranda, E.R., Biles, A.: Evolutionary Computer Music. Springer, Heidelberg (2007)
18. Murray-Rust, D., Smaill, A., Edwards, M.: MAMA: An architecture for interactive musical agents. In: ECAI: European Conference on Artificial Intelligence, pp. 36–40 (2006)
19. Nechvatal, J.: Computer Virus Project 2.0, `http://www.eyewithwings.net/nechvatal/virus2/virus20.html` (accessed 6 October 2010)

20. Schnell, N., Borghesi, R., Schwarz, D., Bevilacqua, F., Müller, R.: FTM – Complex Data Structures for Max. In: Proceedings of the ICMC 2005, Barcelona (2005)
21. Schwarz, D.: Corpus-based concatenative synthesis. IEEE Signal Processing Magazine 24(2), 92–104 (2007)
22. Wooldridge, M.: An Introduction to multiagent systems. Wiley & Sons, Chichester (2009)
23. Wulfhorst, R., Flores, L., Alvares, L., Vicari, R.: A multiagent approach for musical interactive systems. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 584–591. ACM Press, New York (2003)
24. Zwicker, E., Terhardt, E.: Analytical expressions for critical-band rate and critical bandwidth as a function of frequency. Journal of the Acoustical Society of America 68(5), 1523–1525 (1980)

Creating Choreography with Interactive Evolutionary Algorithms

Jonathan Eisenmann[1], Benjamin Schroeder[1], Matthew Lewis[2], and Rick Parent[1]

[1] Computer Science & Engineering, The Ohio State University, 2015 Neil Ave, Columbus, OH, USA
[2] Advanced Computing Center for the Arts & Design, The Ohio State University, 1224 Kinnear Rd, Columbus, OH, USA

Abstract. Directing a group behavior towards interesting and complex motion can and should be intuitive, iterative, and often participatory. Toward this end, we present a choreographic system that enables designers to explore a motion space based on a parametric model of behaviors. Designers may work with the system by moving back and forth through two complementary stages: first, using an evolutionary algorithm to traverse the space of behavior possibilities, allowing designers to emphasize desired kinds of motion while leaving room for an element of the unexpected, and second, using selected behaviors to direct the group motion of simple performing creatures. In the second stage, evolved group motion behaviors from the first stage are used alongside existing high-level parametric rules for local articulated motion.

Keywords: Evolutionary Design, Animation, Interaction Techniques, Choreography, Behavioral Systems.

1 Introduction

In our natural world, the application of a few simple biological rules can bring forth a flourishing array of biodiversity due to the complex nature of the interplay between these basic rules within a dynamic environment. Similarly, the creative synthesis of simple navigational behaviors and agent actions within a simulated dynamic environment can produce a wide range of realistic and interesting complex group motion. However, the same phenomena that allow for such an explosion of diversity actually make any sort of creative directing, such as tuning the parameters which control the blending of these behaviors, an overwhelming task. This problem is compounded by the fact that beauty is, in a sense, like obscenity: we know it when we see it, and our idea of what we like or dislike changes as our level of experience matures through continued exposure. The choreography of group motion is a complex and multifaceted task, one that can benefit greatly from a collaborative process involving both human

© Springer-Verlag Berlin Heidelberg 2011

and machine. In this paper, we explore the use of evolutionary design to add more complex group motion to an existing parametric choreographic animation system. Our long-term hope is to discover theoretical foundations and practical steps for deeper human/machine collaboration in the authoring of group motion.

Specifically, the problem we wish to address is that of providing an intuitive and iterative method for directing the motion of individuals within a group. The dynamic relationships between group members speak volumes to the human psyche about intentions and meaning. These dynamic relationships, when generated by a behavioral system, are, by nature, emergent, and as such, they are difficult to control in a feed-forward sense. We believe that an iterative design process (as long as the iterations are quick) is invaluable in the creation of this kind of complex behavior.

It is worthwhile here to discuss briefly what is meant by "choreography" in this context. The word "choreography" is often used to describe dance or, perhaps, the motion of crowds or fighting pairs. We are indeed interested in these kinds of choreography in this work, but also take an expanded view of the term. In his essay "Choreographic Objects" [4], the choreographer William Forsythe writes that a choreographic object is "a model of potential transition from one state to another in any space imaginable". Choreography, in this view, has to do with organizing systems of changes. Human body choreography is indeed choreography; but the concept of choreography can be applied in many other ways.

In our evolutionary design system, the computer does the book keeping work of simulating individuals (agents) according to sets of rules that can be adjusted in an iterative manner. The computer presents the designer with an array of possible design solutions, and the designer makes creative decisions at each iteration, and will hopefully encounter several surprising ideas along the way that serve to inspire his or her design process.

The dance literature project *Synchronous Objects for One Flat Thing, Reproduced* [5] includes an interactive online "Counterpoint Tool". With this tool, visitors can observe the motion of multi-armed performing creatures and give them broad choreographic direction using a straightforward button-and-slider interface. (See Figure 1) The creatures' motion consists of both articulated arm rotation and navigation around a virtual stage.

The choreographic interface controls an underlying parametric rule system. This system is simple, but can produce surprisingly complex behavior, and the slider interface works well for interactive (and even performative) use. However, the navigational behavior of the creatures is limited; in particular, creatures have no explicit relationship to one another beyond whether they are moving in unison or not. This is in part because it is unclear how to represent a wide range of relational behaviors using a simple parametric interface.

Evolutionary design presents one possible solution to this problem. Using an interactive evolutionary design system, a choreographer could explore a large motion space, selecting elements of the space for use in later interactive "performances". However, part of the interest of the Counterpoint Tool is the ability

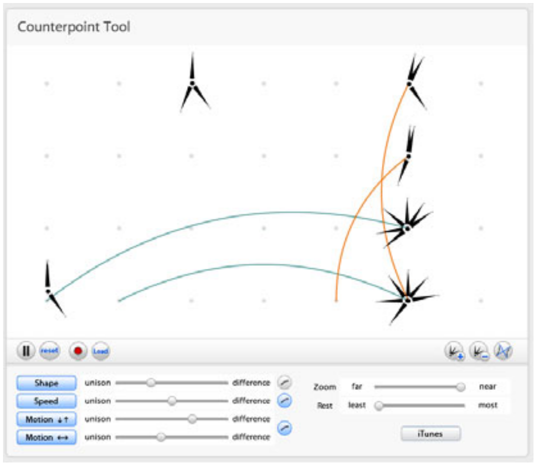


Fig. 1. The original Counterpoint Tool

to give choreographic direction "on the fly". We therefore have worked to combine the two approaches, retaining aspects of high-level, interactive, parametric control while integrating more complex navigational motion designed using behaviors previously created via evolutionary methods.

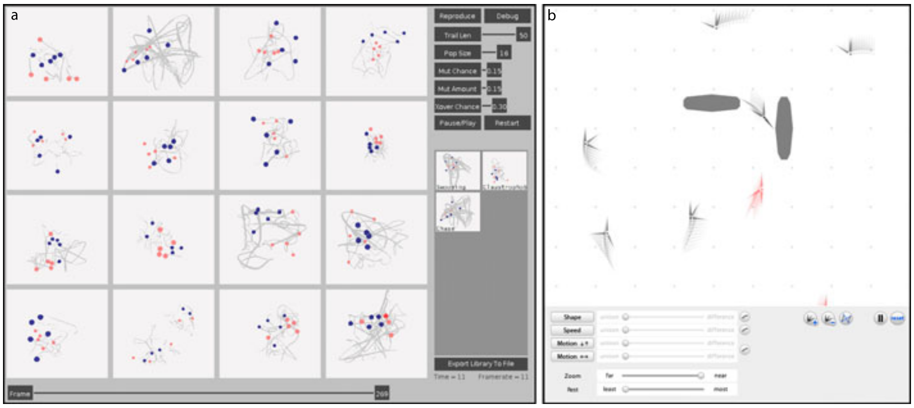
The original Counterpoint Tool was designed with a strong aesthetic target in mind: that of producing visual counterpoint through a system of different kinds of alignments in time. The present work retains the contrapuntal articulated motion of the original tool, but its navigational motion does not have any particular aesthetic aim other than "visual interest" or perhaps behavioral variety as defined by the user.

The original Counterpoint Tool and our expansion of it are choreographic animation systems. Like all behavioral animation systems, they describe the motion of elements related to a space and to each other. However, these tools emphasize the choreographer's role: the creative activity of navigating and shaping a rule space to express artistic ideas.

In this paper we discuss concepts from several fields: behavioral simulation, evolutionary design, parametric modeling, and choreography. We first provide more background on several of these concepts and discuss some related work. Then we describe the particulars of our method, and finally discuss some results from our proof-of-concept system as well as possibilities for future investigation.

2 Foundations

In order to provide some context for our work among the various disciplines from which it draws, we will briefly summarize work within behavioral simulation and interactive evolutionary algorithms, and highlight the concepts that we draw from each.



Trail Len 50
Pop Size 16
Mut Chance 0.15
Mut Amount 0.15
Xover Chance 0.30
Frame 269

Fig. 2. (a) Our interactive evolutionary design interface showing group motions with trails. (b) Our performative interface with moving three-armed creatures.

First, we will tackle the field of behavioral simulation, a fascinating area of study located at the crossroads of artificial life, psychology, and simulation. More specifically, we are interested in using agent (or individual) based models for the simulation of groups [13]. In this way, simple rules applied at the local level can produce emergent or seemingly cooperative/competitive behavior between agents in groups. Significant works include Reynolds's "Steering Behaviors for Autonomous Characters" which we will discuss at greater depth later on [14]. For an excellent overview of simulations involving agents in a dynamic environment, we refer the reader to Dorin's survey on virtual ecosystems in generative electronic art [2]. Perlin's IMPROV system also deals with not only autonomous agents, but also how to direct them interactively [12]. Several notable studies in the area of directable group/crowd motion include papers by Kwon [9], Ulicny, et al. [18], Kim, et al. [8], and Anderson et al. [1].

Next we discuss evolutionary algorithms, another area of artificial intelligence, whose applications seem endless due to their inherent flexibility. We will mention a few works that have inspired us to use interactive evolutionary methods for choreography design. Sims played a pivotal role in establishing these techniques as standards in the graphics literature. Application areas explored by Sims include texture synthesis [15], particle dynamics [16], and evolved virtual creatures [17]. Lim & Thalmann [11], Ventrella [19], and Eisenmann et al. [3] have done some work with evolving character animation, and Li & Wang have applied non-interactive evolutionary algorithms to generate optimized crowd movement [10]. Another work relevant to ours is Jacob and Hushlak's discussion of boid-like evolved swarm choreography [7]. Our goal is to add to the body of knowledge in this area by providing a way to interactively evolve group motion based on steering behavior rules.

3 Our Method

3.1 Agent Navigation

The navigational motion is synthesized from basic building blocks called steering behaviors. The behavior building blocks used by the sytem include: freeze, arrive, pursue/evade, follow, wander, explore, hide, contain, and separate. Most of these steering behaviors have additional parameters like vision variables or a proximity radius. For more details, we refer the reader to Reynolds's work [14], from which we drew heavily, extending and making slight changes as we saw fit.

The only behaviors which we have added to Reynold's list are freeze, explore, and hide. The freeze behavior applies a braking force in the direction opposite to the agent's velocity. The explore behavior uses the arrive behavior to navigate through a list of goals scattered across the space. The hide behavior steers an agent toward the point on the nearest obstacle that is furthest from the nearest agent pursuing the agent in question.

Each behavior applies a force to the agent, steering towards or away from a particular agent, goal, or obstacle. In our implementation, a weighted sum of these forces provides the steering force for an agent. The agents are modeled similarly to the agents in Reynold's method, with a minimal set of properties that define physical limitations such as maximum speed and acceleration.

Choosing the correct weights for these forces is a tricky business. If too many behaviors are blended together, the resulting motion appears non-intentional and indecisive most of the time. The effect is much like that of mixing colors. If a painter mixes all the colors on his or her palette together, the result is usually an uninteresting muddy color. However, if a few colors are mixed in the right ratios, diverse and interesting hues can result. Furthermore, if these composite colors are delivered to the canvas in an interesting rhythm and at the proper points in 2D space, a work of art is born. Extending this metaphor, we would like our system to not only create interesting mixtures of behaviors, but also to deliver them to the viewer at distinct points on the canvas of space and time.

To this end, our system includes a set of state machines that govern the behavior of each agent. Each state stores a set of weights so that the system can create a blended behavior for that state by calculating a weighted sum of all possible steering forces. In the interest of keeping the number of system variables as low as possible, we use a three state fully connected graph to represent the behavior of all the agents in one group. The weights of the states and transitions as well as the target information are evolved separately for each group of agents. At each simulation step, an agent will check its state machine to see which state it is currently in, and then will use the corresponding state weights to sum up its steering forces. We apply a low-pass filter to the steering forces before they are summed so that we only include the strongest signals and thus avoid mixing too many behaviors at once.

We will now examine the game of hide-and-seek to gain some practical context for what these states and transitions could be. Hide-and-seek is a subset of the possible behaviors in our system, and it requires only two groups of agents:

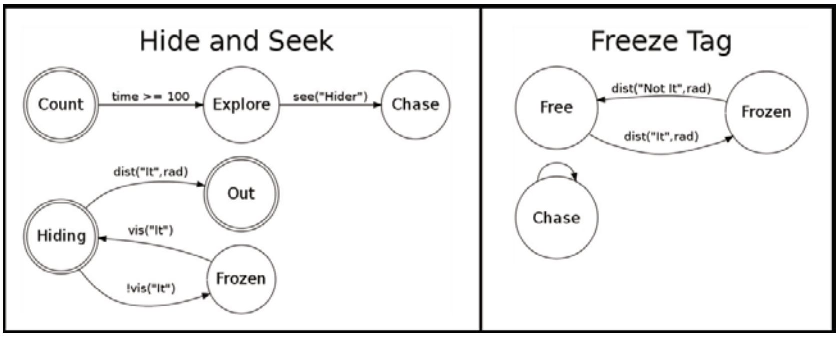


Fig. 3. Two finite state diagrams showing two possible behaviors contained within our generalized behavior model

the hiders and the seeker(s). Each set of agents will have their own state machine definition. The hiding agents would start in a behavior state that consists primarily of the hiding and evasion behaviors, but also includes the separation and containment behaviors, so that they do not run into each other and also so they stay within the play area and do not run into obstacles in the environment. When these agents sense that they are invisible to “It” they transition to a frozen state, until they sense that they are visible to “It”. When this occurs, they return to the original hiding/evasion state. If they sense that they are within tagging proximity to “It” they will go into a wander state, and leave the play area because they have been tagged “out”. The “It” agent will begin in a frozen state that signifies counting to 100. “It” will leave this state after a duration of 100 seconds, and begin to explore the play area. When “It” is able to see a hiding agent, he moves into a pursuit state (which also includes separation and containment). When an agent is tagged, that agent is removed from “Its” target list. (See figure 3 for the finite state diagram of this game as well as the diagram for the game of freeze tag.) Each state stores the weights to be used in the sum of steering forces, and also any pertinent target information (i.e. evade agents in subset X of group Y). Note that this state data is not shown in figure 3. There are six possible transitions between states. The transitions correspond to events or changes in attributes of the system. An agent’s state machine will move to a new state given a probability equal to the weight for the transition in question, but only if the condition specified by the type of transition evaluates to true. Transition types include:

– proximity to targets
– visibility of self from targets
– invisibility of self from targets
– ability to see targets
– inability to see targets
– time in state surpasses a specified duration

3.2 Interactive Evolutionary Algorithm

We use interactive selection from the user at each generation in the first stage of our system to determine the fitness of our group motion phenotypes. The genotype is a fixed-length array of floating point numbers corresponding to steering behavior parameters, agent properties, state weights, and transition weights for each group of agents. The initial population is randomly generated, and we allow the user to introduce genes from a library of examples at any time. In each generation, the user's selections form a pool of parents available to the mating algorithm for reproduction. The mating algorithm chooses two distinct, random parents from this pool for each new offspring. We use a probabilistic crossover where the probability that the algorithm starts copying from the other parent at each gene is controlled by the user. Mutation takes place after crossover by randomly changing gene values by adding a random value between -1 and 1 that is scaled by a mutation amount. Mutation rate and mutation amount can be adjusted by the user as well. We use elitism, and population size can be set by the user at any time.

In our current prototype, the user generates as many interesting behaviors as desired using the interactive evolutionary design interface. Group motion of particular interest to the user is then stored in a library. The library also contains some classic examples from children's games such as hide-and-seek and freeze tag. The motions stored in this library can be utilized later in the evolutionary process as a means of (re)seeding the generation with a particular desired behavior.

In a typical design session, the user often views 16 individuals at a time and spends a minute or two with each generation. The examples displayed in our results were produced in anywhere from 9 to 12 generations.

3.3 User Interface Concerns

One technique that we found helpful for dealing with the well-known user fatigue problem in interactive evolutionary computation, especially in the case of time-varying data, was to display trails (or traces) of the agents as they move through space and time (Figure 2 on the left). The position of the agents is recorded every so often and a spline is drawn through these stored points. The thickness of the spline correlates to the speed of the agent at that point in time. In this way, the time-varying data can be compressed into a single-image at any frame of the animation. The length of the trails is controllable by the user so that he or she can determine how much history is visible in any one frame. We note that this style of drawing agent path history may, in some cases, appear similar to ant drawings. However, it is actually a visual compression technique applied to spatio-temporal data. Furthermore, the underlying navigational rules defining the system do not have anything to do with the notion of pheromone trails.

As mentioned earlier, our interface involves two separate stages. Designers first use an interactive evolutionary design interface to explore the space of possible motions, saving some in a library for later use. They then use the library in conjunction with a "performative" interface similar to that of the Counterpoint Tool. In the performative interface, users may mix and match the original parametric Counterpoint Tool behaviors and the saved library of evolved behaviors to create new motion.

In the interest of maintaining visual simplicity within the interactive evolutionary design stage of our system, we have chosen to use simple shapes to represent our agents and a two dimensional environment for them to live in. We decided that it was not necessary to use a more complex representation, because we wanted the amount of visual information presented to the user at any one time to be kept to a minimum. A study by Heider & Simmel has shown that even simple shapes can evoke sympathy and communicate intentions to a human audience [6]. However in the performative stage of our system we wish to enhance the visual complexity of the agent motions, so we add procedurally generated arm movements and layer them on top of the agents' navigational behaviors.

Because we believe it is important to be able to make some decisions during a performance, we have retained the high-level shape and speed sliders from the original Counterpoint Tool. These sliders control the arm movement model. The arm movements add considerable visual interest, although they are not evolved and do not interact directly with the navigational motion.

The Counterpoint Tool also has a "rest" control, which directs the creatures to pause more or less often, regardless of the underlying navigational algorithm. This one control value is connected both to the articulated arm movement and also to the steering behavior weighting. In the case of articulated arm movement, the "rest" value controls the percentage of the time that the arms are still. In the case of navigational behaviors, the value determines with what probability other behaviors (besides the freeze behavior, which stops the agents, and the separate and contain behaviors, which keep agents from running into things) will be ignored. This control over stillness can change the look and feel of a group's motion dramatically, providing considerable visual interest and variety.

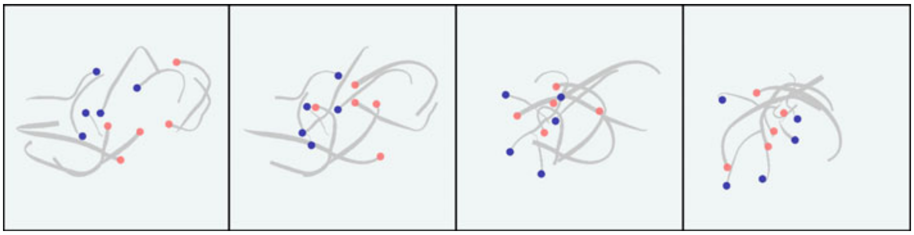


Fig. 4. A sequence of images from one of the behaviors evolved in the first stage of our system. In this example the speedy red agents are harrassing the sluggish blue agents, and breaking up their cluster.

4 Results

The best way to view motion results is often through seeing the motion itself. We have therefore posted videos of a few example behaviors to our project website at: http://accad.osu.edu/Projects/Evo/Choreography/. There is also a video that documents the results of an interactive choreography session, demonstrating the full range of the performative interface.

See figure 4 for a filmstrip showing a sequence of thumbnails of a behavior evolved in the first stage of our system. The images in Figure 2 show screenshots of the two stages of our system. In the performative interface image (figure 2b), the creatures have been assigned a particular behavior that was created with the interactive evolutionary design interface (figure 2a), saved to the library, and rendered here along with the articulated arm motion. Note that we have added motion blur to this image to give the feeling of movement. The two polygons in the environment are obstacles that the creatures interact with and can hide behind if being pursued.

5 Future Work

In the future we plan to move our two stages into a more seamless interface so that users can move back and forth easily between evolutionary design of navigation behaviors and high-level parameter adjustment in the performative part of the system. We would also like to experiment with a more collaborative and immersive interface. This interface would consist of a touch table control panel and a virtual environment projected onto the floor where users could participate with the behavior at hand by controlling a free agent through computer vision techniques.[1] We feel this might lead to collaborative synergy between two users: one at the touch table navigating the parametric behavior space, and the other participating in the choreography. Additionally, we would like to investigate the use of automatic filters for distribution, dynamism, and balance that the designer could use to customize the sampling of motion that is presented in each generation. We would also like to experiment further with the parametric space we have created in order to learn what (if any) combinations of genes correspond to high-level controls. Currently our only such high-level control is for "rest", but we would like to develop more controls like this.

6 Conclusion

In conclusion, we have taken some exciting first steps into the realm of parametric choreographic group motion using evolutionary design techniques. The results produced with our prototype system show great promise, and we hope that our future work will lead to both new ways to interact with and learn from choreographic ideas as well as better ways to explore and author new group motion sequences.

[1] Zuniga Shaw et al. used a similar interface in their recent installation *Synchronous Objects, reproduced* [20].

References

1. Anderson, M., McDaniel, E., Chenney, S.: Constrained animation of flocks. In: SCA, pp. 286–297 (2003)
2. Dorin, A.: A survey of virtual ecosystems in generative electronic art. In: The Art of Artificial Evolution: a Handbook on Evolutionary Art and Music, pp. 289–309 (2008)
3. Eisenmann, J., Lewis, M., Cline, B.: Interactive evolutionary design of motion variants. In: International Conf. on Evolutionary Computation (2009)
4. Forsythe, W.: Choreographic Objects (2009), http://www.wexarts.org/ex/forsythe (accessed)
5. Forsythe, W., Palazzi, M., Zuniga Shaw, N., et al.: Synchronous Objects for One Flat Thing, Reproduced (2009), accessed at www.synchronousobjects.osu.edu
6. Heider, F., Simmel, M.: An Experimental Study of Apparent Behavior. The American Journal of Psychology, 243–259 (1944)
7. Jacob, C., Hushlak, G.: Evolutionary and swarm design in science, art and music. The Art of Artificial Evolution, 145–166 (2008)
8. Kim, M., Hyun, K., Kim, J., Lee, J.: Synchronized multi-character motion editing. ACM Trans. on Graphics, 79:1–79:9 (2009)
9. Kwon, T., Lee, K.H., Lee, J., Takahashi, S.: Group motion editing. In: SIGGRAPH, pp. 1–8 (2008)
10. Li, T.Y., Wang, C.C.: An evolutionary approach to crowd simulation. In: Autonomous Robots and Agents, pp. 119–126 (2007)
11. Lim, I.S., Thalmann, D.: Pro-actively interactive evolution for computer animation. In: Computer Animation and Simulation, pp. 45–52 (1999)
12. Perlin, K., Goldberg, A.: Improv: A System for Scripting Interactive Actors in Virtual Worlds. In: SIGGRAPH, pp. 205–216 (1996)
13. Reynolds, C.: Individual-Based Models (2010), www.red3d.com/cwr/ibm.html (accessed)
14. Reynolds, C.: Steering Behaviors For Autonomous Characters. In: Game Developers Conf., pp. 763–782 (1999)
15. Sims, K.: Artificial evolution for computer graphics. In: SIGGRAPH, pp. 319–328 (1991)
16. Sims, K.: Particle animation and rendering using data parallel computation. In: SIGGRAPH, pp. 405–413 (1990)
17. Sims, K.: Evolving 3d morphology and behavior by competition. Artificial Life 1(4), 353–372 (1994)
18. Ulicny, B., Ciechomski, P., Thalmann, D.: Crowdbrush: interactive authoring of real-time crowd scenes. In: SCA, pp. 243–252 (2004)
19. Ventrella, J.: Disney meets darwin-the evolution of funny animated figures. In: Woodbury, R., Williamson, S., Beesley, P. (eds.) Computer Animation (1995)
20. Zuniga Shaw, N., et al.: Synchronous Objects, reproduced (art installation). ISEA, Essen, Germany (2010)

Modelling Human Preference in Evolutionary Art

Anikó Ekárt[1], Divya Sharma[2], and Stayko Chalakov[1]

[1] Aston University
Aston Triangle, Birmingham B4 7ET, UK
a.ekart@aston.ac.uk, stayko16@yahoo.com
[2] IIT Ropar, India
divyas@iitrpr.ac.in

Abstract. Creative activities including arts are characteristic to humankind. Our understanding of creativity is limited, yet there is substantial research trying to mimic human creativity in artificial systems and in particular to produce systems that automatically evolve art appreciated by humans. We propose here to model human visual preference by a set of aesthetic measures identified through observation of human selection of images and then use these for automatic evolution of aesthetic images.

Keywords: aesthetic measure, human preference modelling, genetic programming, interactive vs automatic evolution.

1 Introduction

Ever since the invention of the first computing device, humanity has been thinking about using them to perform creative activities. Producing aesthetically pleasing pieces of art is certainly one such creative activity. Beginning with the pioneering work of Dawkins [7] and Sims [19], over the past twenty years a lot of effort has been spent on generating increasingly more effective evolutionary art systems that produce aesthetic artworks. Successful examples attracting substantial public attention include the Electric Sheep [9], the NEvAr system [14] and the Painting Fool [3].

The majority of evolutionary art systems are either interactive (for example [21]) or automatic (for example [1,8]). Interactive systems tend to generate more aesthetic artworks, as their driving force is human selection, but at the same time need a lot of effort on the part of the human, may incur user fatigue and could be inconsistent over time. Automatic systems have the advantage of a built-in automatic fitness evaluation, so the human effort is reduced; however, the aesthetics of the resulting artworks may suffer as the automatic evaluation has not been perfected yet. To overcome the disadvantages and also combine the advantages of both approaches, Machado et al. propose partially interactive evolution [15], where the human user's contribution is much reduced compared to the fully interactive approach, but the human still guides the evolution.

© Springer-Verlag Berlin Heidelberg 2011

Substantial efforts in evolutionary art research have been dedicated to studying and devising good aesthetic measures [11,13,17,18]. It is generally agreed that formulating a universally valid and acceptable aesthetic criterion is not within our reach. Achieving automatic evolution that produces aesthetic images to the liking of the human user very strongly depends on the understanding of the particular user's aesthetic values. A recent study by Li and Hu [12] suggests using machine learning to learn the differences between aesthetic and non-aesthetic images, as indicated by image complexity and image order. Colton [4] produces new rules for forming fitness functions through the use of an inference engine. Greenfield proposes the technique of evolutionary refinement [10] to encourage aesthetic pattern formation through stages and concludes that "evolution in stages with radical changes in fitness criteria may be a profitable evolutionary exploration strategy".

Our contribution complements these previous approaches by considering four established aesthetic measures in interactive evolutionary art to model human preference. We monitored how these measures evolved over the generations when different users interacted with a simple evolutionary art system and fully drove the selection process. We found that a combination of aesthetic measures models user preference suitably well. We consequently employed this combination (MC and BZ) to automatically evolve further images starting from the result of interactive evolution.

2 Aesthetic Measures

We study the evolution of four well-known aesthetic measures in an attempt to model human selection in interactive evolutionary art. Measure R is based on Ralph's work, measure MC is based on Machado and Cardoso's work, measure BZ on Birkhoff and Zurek's work and finally measure S on Shannon entropy.

2.1 Aesthetic Measure R

This aesthetic measure is based on the mathematical model proposed by Ralph[18]. After analyzing hundreds of examples of fine art, it was found that many works consistently exhibit functions over colour gradients that conform to a normal or bell curve distribution.

The colour gradient for each pixel is computed as:

$$|\nabla r_{i,j}|^2 = \frac{(r_{i,j} - r_{i+1,j+1})^2 + (r_{i+1,j} - r_{i,j+1})^2}{d^2}$$

where $r_{i,j}$ is the value of the Red component for pixel (i, j) and d is a scaling factor which is taken to be 0.1% of the diagonal length, as suggested by Ralph's model [18] (leading to the value $d^2 = 3.277 * 10^{-2}$). $\nabla g_{i,j}$ and $\nabla b_{i,j}$ are computed similarly for the Green and the Blue colour components.

The overall gradient, or the stimulus, of each pixel is calculated as follows:

$$S_{i,j} = \sqrt{|\nabla r_{i,j}|^2 + |\nabla g_{i,j}|^2 + |\nabla b_{i,j}|^2}.$$

Next, the viewer's response to each pixel is computed as

$$R_{i,j} = \log(S_{i,j}/S_0).$$

The range of values for $R_{i,j}$ is $[0, 1)$. R can never become negative. The minimum value of 0 corresponds to the case when there is no change in colour at a pixel at all; if there is no stimulus the response is 0. $S_{i,j}$ can never be less than S_0 due to the scaling factor. S_0 is the detection threshold taken to be 2, as suggested by Ralph's model of aesthetics. If $S_{i,j} = 0$ (no change in colour at a pixel), it is ignored.

The mean μ and standard deviation σ of the response values are calculated using the response values themselves as weights because the probability that a viewer pays attention to a detail of an image is considered proportional to the magnitude of the stimulus that resides at that detail. A histogram is built next to judge how close the distribution of response values is to the bell curve distribution. The "bins" will each represent an interval of size $\sigma/100$. Then the probability that a response value falls in a given bin is computed. This is repeated for all the bins by going through all the $R_{i,j}$ values where each $R_{i,j}$ updates its corresponding bin using a weight of $R_{i,j}$.

Then, the deviation (D) from the normal distribution is computed as follows:

$$D = \sum_i p_i \log\left(\frac{p_i}{q_i}\right)$$

where p_i is the observed probability in the i^{th} bin of the histogram and q_i is the expected probability assuming a normal distribution around μ with standard deviation σ. When $q_i = 0$, that bin is ignored. The value $e^{-|D|}$ will reported as the value of the aesthetic measure. With a value between 0 and 1, a low value will indicate a large deviation and hence a poor image, whereas a large value will correspond to a good image.

We justify this aesthetic measure as follows:

1. Aesthetic measure R discourages images which give rise to very high or very low response values. If a viewer gives very little response to something, it is too insignificant to be of interest. On the other hand, if a viewer gives a very large response to something, it is too disturbing or chaotic.
2. The response value increases as the gradient increases and decreases as the gradient falls. Very low gradients give rise to single coloured monotonous areas (which do not interest a viewer) whereas very large gradients give rise to sharp lines and boundaries separating areas with huge colour differences (which is undesirable). Aesthetic measure R discourages very high and very low gradients and encourages reasonable values of gradients.

2.2 Aesthetic Measure MC

This measure is based on the aesthetic theory of Machado and Cardoso [13] asserting that the aesthetic value of an artwork is directly connected to Image Complexity (IC) and inversely connected to Processing Complexity (PC). So, the value of the aesthetic measure is calculated as the ratio

$$\frac{IC}{PC}. \tag{1}$$

In order to compute IC, we first compress the image losslessly using JPEG compression and calculate the ratio (I) of the size of compressed image to the size of uncompressed image. We hypothesize that the IC is directly connected to the ratio I. The inherent unpredictability, or randomness can be measured by the extent to which it is possible to compress the data [6]. Low values of I indicate substantially compressible and low complexity image. High values of I indicate not very compressible and therefore more complex image. That is,

$$\text{more compressible} \equiv \text{less random} \equiv \text{more predictable} \equiv \text{less complex}$$

Hence, the less the value of ratio I (the less the size of the compressed file) is, the more compressible and hence, the less complex the image is. We substitute the ratio I for IC in Equation 1.

PC should reflect the complexity of the coding of the image. We encode each image by three expression trees, one for each of the R, G and B components, as detailed in Section 3. In order to compute PC, we compress the expression trees represented as strings in prefix notation and again find the ratio P of size after compression to size before compression. We argue that PC can be substituted by the ratio P. The aesthetic measure MC will be computed as

$$\frac{I}{P}.$$

In theory the value of this aesthetic measure could range from zero to infinity, where infinity corresponds to an image that cannot be compressed, but whose genetic expression tree can be compressed to the minimum. Zero corresponds to an image that can be compressed significantly, but with an expression that cannot be compressed. It is notable that the compression rate PC of the mathematical expressions could be replaced with the more exact rate computed for the minimum length of an equivalent mathematical expression. However, as arithmetic simplification is not applied on the mathematical expressions in our system, we consider that using the actual evolved expression is appropriate.

2.3 Aesthetic Measure BZ

This aesthetic measure is based on Birkhoff's measure [2] and Zurek's physical entropy [17]. We compute the value of Shannon's entropy as mentioned in [17] by creating a histogram of luminance values of pixels and computing Shannon's entropy H_p as follows:

$$H_p = -\sum_i p_i \log p_i$$

where p_i is the probability in the i^{th} bin of the histogram. The luminance value (L) for a pixel (i, j) is computed as follows:

$$L = (0.2126 * r_{i,j}) + (0.7152 * g_{i,j}) + (0.0722 * b_{i,j}).$$

Next, the Kolmogorov Complexity (K) [17] of the expression trees of the image is estimated by compressing the strings corresponding to expression trees and finding the length of the compressed string. The value of this aesthetic measure is given by

$$\frac{H_p}{K}$$

This aesthetic measure discourages very high and very low luminance values because it favours high values of H_p. Very high and very low luminance values lead to low values of H_p. Here, K is used as a measure of PC.

2.4 Aesthetic Measure S

As stated in [17], to analyse an image's composition, the used measures must quantify the degree of correlation or similarity between image parts. We compute this degree of correlation by dividing the image into four equal squares and compute Shannon's entropy (H_{p_i}, $i = 1, \ldots, 4$) for each of these parts. We then compute the weighted mean of these values (the weight being the area of the part). Finally, we find the ratio of the weighted mean to the Shannon's entropy value of the image as a whole to obtain the value of the aesthetic measure. The value of the aesthetic measure is given by

$$\frac{H_{p_1} + H_{p_2} + H_{p_3} + H_{p_4}}{4H_p}.$$

3 The Underlying Evolutionary Art System

A simple interactive evolutionary art system is used, where the user is presented with nine images and has to select two as parents for the next generation. Images are represented by triplets of functions corresponding to the three components R, G and B. For each pixel of the image, the values of these functions are calculated and produce the colour of the pixel as shown in Fig. 1.

Genetic programming is employed for the evolution of the expression trees. Simple subtree crossover, subtree and point mutation are the allowed genetic operators. The user can set the operators to be used and their rates and can also introduce new random images at anytime during the interactive evolution. The system was implemented in Java and uses all mathematical functions provided in Java.Math. The terminals are Cartesian and polar coordinates of the points and also random constants.[1] Examples of images produced by the authors are shown in Fig. 2.

4 Simple Numerical Analysis

To see whether there are any particular functions that are preferred more than others, we initially analysed their frequency in nice images. The number of occurrences of each function in 44 nice images manually selected from a set of images generated through interactive evolution by the authors in sessions of

[1] The interactive system is available for download at http://www.evoartmedia.com

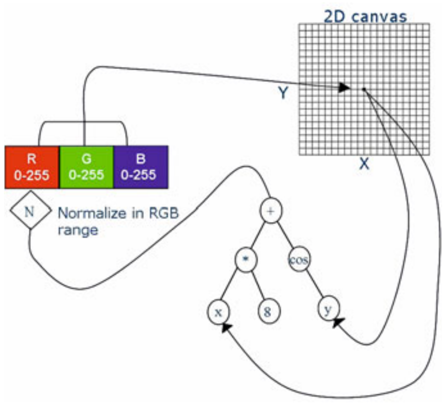
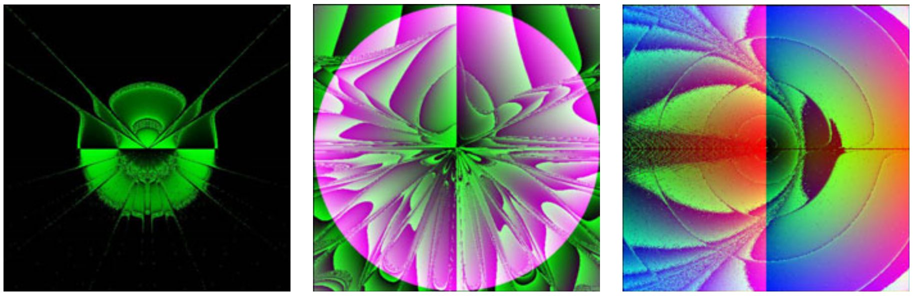


Fig. 1. The genetic representation used



(a) Spider (b) Flower (c) Fish

Fig. 2. Example images generated by interactive evolution

length varying between 15 and 30 minutes is shown in Fig. 3. The images were selected such that on visual inspection they looked substantially different.

It can be seen that the preferred functions are SEC, CUBRT, EXP, LOG, SQRT +, -, MAX, AVG, *, as each of these occurs on average at least once in every aesthetic image. At 63.8% of all variables, polar coordinates were the preferred variables and variables were preferred over numeric constants, as 73.3% of terminals. The constant range $[0.5, 0.6)$ had the highest frequency of 20% of all constants, the next closest being $[0.6, 0.7)$ at 12.5%. Such a simple analysis does not really offer detailed understanding of human aesthetic selection, just indicates the more likely ingredients of aesthetic images within the given class of images.

5 Aesthetic Measures to Model Human Selection

Individual interactive evolution experiments were analysed to better understand their driving force: human selection. Then automatic evolution was applied to the resulting image, using the selection criteria revealed by the analysis. The analysis involved monitoring the evolution of the four aesthetic measures described in

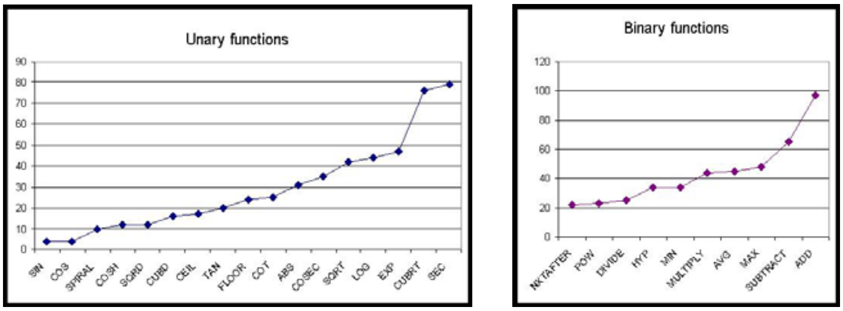
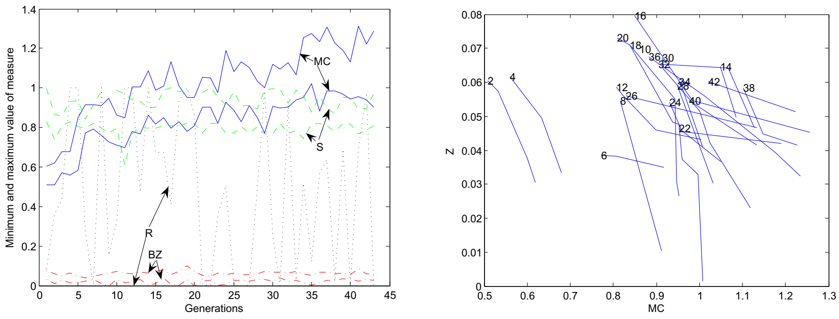


Fig. 3. Occurrences of various functions in 44 different aesthetic images obtained in different runs of the system



(a) Evolution of measure values (b) Pareto front (MC,BZ)

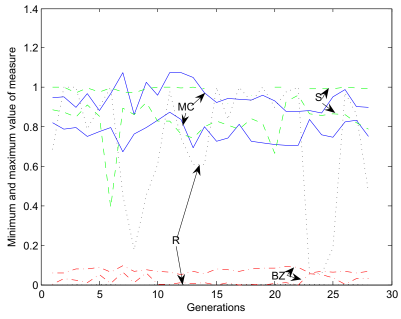
Fig. 4. Evolution of image through interactive evolution by computer scientist

Section 2 during interactive experiments performed by four different people. We found that although there are similarities for all users, the occupation of the user substantially influences their use of the system and their image selection preference.[2] There is no universally applicable simple model. Computer scientists tend to use the system for longer than graphic designers. The value of measure MC shows a clear growth over generations in the case of computer scientists, as in Fig. 4(a), both for the minimum and the maximum value taken in each generation. At the same time, there is no clear tendency in the evolution of values for measure MC in the case of graphic designers (see Fig. 5(a)). The evolution of measures BZ and S are similar for both types of users: variation within similar ranges is observed, approximately $[0, 0.05]$ for BZ and $[0.8, 1]$ for S, respectively. The R measure has a lot of variation across its full range for computer scientists and somewhat less variation for graphic designers, but follows no clear pattern.

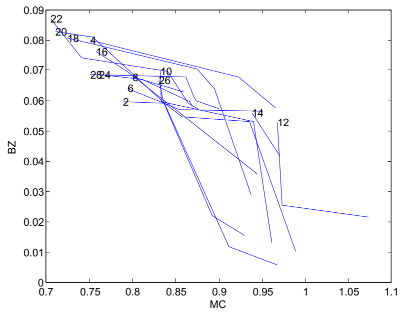
Interestingly, if we consider the evolution of two measures together and draw the Pareto front of non-dominated solutions[3] in each generation, we notice some

[2] The users were computer scientists and graphic designers.

[3] A point (x_1, y_1) is part of the Pareto front of a set if there is no other point (x_2, y_2) in the same set such that $x_2 > x_1$ and $y_2 > y_1$.

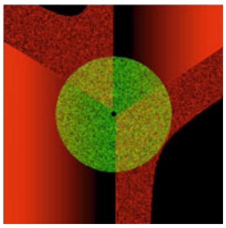


(a) Evolution of measure values



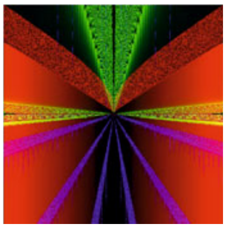
(b) Pareto front (MC,BZ)

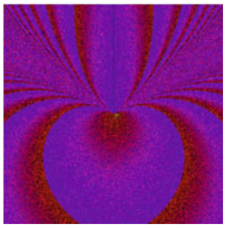
Fig. 5. Evolution of image through interactive evolution by graphic designer

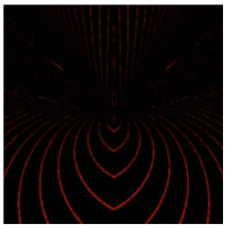


R = 3.8E-33
MC = 0.29
BZ = 0.04
S = 0.82

(a) Computer scientist

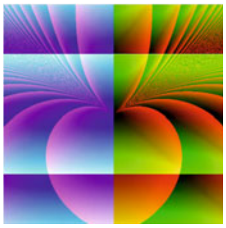

(b) Automatic mut.


(c) Automatic Xover



R = 0.47
MC = 0.26
BZ = 0.015
S = 0.99

(d) Graphic designer


(e) Automatic mut.

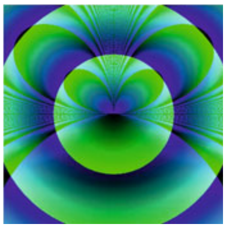

(f) Automatic Xover

Fig. 6. Images created by computer scientist and graphic designer. Subsequent images evolved from these by automatic evolution using mutation or crossover.

trends. Figures 4(b) and 5(b) show the evolution of the Pareto front for measures MC and BZ. In both shown examples, with a few exceptions over the full experiment, the front is moving toward better (i.e. higher values of aesthetic measures) non-dominated solutions. We interpret this as an indication that human users may not select the images during evolution in a way that consistently follows a single aesthetic measure, but more likely a set of aesthetic measures. In fact if we compare the two images in Fig. 6(a) and 6(d), we notice that the first image scores better over measures MC and BZ, while the second image scores better over measures R and S.

When attempting partial automatic evolution [15,20] we propose that the human's previous selections are analysed and modelled by the best fitting set of measures and then the automatic evolution subsequently uses these measures. It is then more likely that images to the particular human user's liking are produced by automatic evolution. We therefore applied automatic evolution with the combination of the MC and BZ fitness measures to create images starting from the preferred images of the human users. We experimented with mutation only or both crossover and mutation, various settings of population sizes (15-40) and generation numbers (30-100) allowing the computer to spend different amounts of time on creating new images. Evolved images are shown in Figures 6(b) 6(c) and 6(e), 6(f), respectively.

6 Conclusion

We proposed modelling human user preference by a set of aesthetic measures monitored through observation of human selection in an interactive evolutionary art system. Although our evolutionary art system is very simple and is only capable of generating images within a limited set, it provides a suitable environment for studying human aesthetic judgment. The same principles could be applied using an extended set of aesthetic measures on more sophisticated evolutionary art systems and then different combinations of aesthetic measures may be found to model individual users best.

McCormack [16] criticises aesthetic selection itself and proposes an open problem "to devise formalized fitness functions that are capable of measuring human aesthetic properties of phenotypes". The key is to *model and measure* human aesthetic properties by the available means.

We argue that once a combination of measures that models increasing human preference during interactive evolution is identified, automatic evolution is provided with a suitable fitness evaluation method. We are planning to conduct more experiments using an approach similar to that of Colton et al. [5]. Also we are planning to employ machine learning techniques to find potentially more accurate functions driving human aesthetic judgment and to subsequently apply these functions for evaluation and selection in automatic evolution.

Acknowledgments

The authors acknowledge the support of Aston University's summer internship scheme which made it possible for the second author to take part in the reported work. The authors are grateful to Philip Welch, Ken Murphy, Adrian Carolan and Daniel Baxendale for taking part in the user experiments.

References

1. Baluja, S., Pomerleau, D., Jochem, T.: Towards Automated Artificial Evolution for Computer-generated Images. Connection Science 6, 325–354 (1994)
2. Birkhoff, G.D.: Aesthetic Measure. Harvard Univ. Press, Boston (1933)

3. Colton, S.: The Painting Fool (2007), www.thepaintingfool.com
4. Colton, S.: Automatic Invention of Fitness Functions with Applications to Scene Generation. 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.Ş., Yang, S. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 381–391. Springer, Heidelberg (2008)
5. Colton, S., Gow, J., Torres, P., Cairns, P.: Experiments in Objet Trouvé Browsing. In: Conference on Computational Creativity, pp. 238–247 (2010)
6. Dasgupta, S., Papadimitriou, C., Vazirani, U.: Algorithms. Tata McGraw-Hill Publishing Company Limited, New York (2008)
7. Dawkins, R.: The Blind Watchmaker (1986)
8. DiPaola, S., Gabora, L.: Incorporating Characteristics of Human Creativity into an Evolutionary Art System. Genetic Programming and Evolvable Machines 2(1) (2009)
9. Draves, S.: Electric Sheep (1999), www.electricsheep.org
10. Greenfield, G.: Generative Art and Evolutionary Refinement. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 291–300. Springer, Heidelberg (2010)
11. den Heijer, E., Eiben, A.E.: Comparing Aesthetic Measures for Evolutionary Art. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 311–320. Springer, Heidelberg (2010)
12. Li, Y., Hu, C.J.: Aesthetic Learning in an Interctive Evolutionary Art System. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 301–310. Springer, Heidelberg (2010)
13. Machado, P., Cardoso, A.: Computing Aesthetics. In: de Oliveira, F.M. (ed.) SBIA 1998. LNCS (LNAI), vol. 1515, pp. 219–228. Springer, Heidelberg (1998)
14. Machado, P., Cardoso, A.: All the Truth about NEvAr. Applied Intelligence, Special Issue on Creative Systems 16(2), 101–119 (2002)
15. Machado, P., Romero, J., Cardoso, A., Santos, A.: Partially Interactive Evolutionary Artists. New Generation Computing 23, 143–155 (2005)
16. McCormack, J.: Open Problems in Evolutionary Music and Art. 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. 428–436. Springer, Heidelberg (2005)
17. Rigau, J., Feixas, M., Sbert, M.: Informational Aesthetics Measures. IEEE Computer Graphics and Applications, 24–34 (2008)
18. Ross, B.J., Ralph, W., Zong, H.: Evolutionary Image Synthesis Using a Model of Aesthetics. In: IEEE Congress on Evolutionary Computation, pp. 1087–1094 (2006)
19. Sims, K.: Artificial Evolution for Computer Graphics. In: SIGGRAPH 1991, vol. 25, pp. 318–328 (1991)
20. Todd, S., Latham, W.: Evolutionary Art and Computers. Academic Press, London (1992)
21. Ventrella, J.J.: Evolving the Mandelbrot Set to Imitate Figurative Art. In: Hingston, P.F., Barone, C.L., Michalewicz, Z. (eds.) Design by Evolution, pp. 145–167 (2008)

Evolution of Architectural Floor Plans

Robert W.J. Flack and Brian J. Ross

Dept of Computer Science, Brock University,
St Catharines ON L2S 3A1, Canada

Abstract. Layout planning is a process of sizing and placing rooms (e.g. in a house) while attempting to optimize various criteria. Often there are conflicting criteria such as construction cost, minimizing the distance between related activities, and meeting the area requirements for these activities. This paper describes new techniques for automating the layout planning process using evolutionary computation. New innovations include allowing polygonal exteriors and multiple floors. Multi-objective ranking algorithms are tested to balance the many objectives in this problem. The evolutionary representation and requirements specification used provide great flexibility in problem scope and depth of problems to be considered. A variety of pleasing plans have been evolved with the approach.

Keywords: evolutionary design, floor planning, genetic algorithms, multi-objective optimization, Pareto ranking, ranked sum.

1 Introduction

Architecture is a complex amalgamation of science and art. There are functional requirements, cultural expectations and general guidelines to follow. But within these guidelines, there are still limitless possibilities. Even though a house may meet building codes and social norms, Hirsch feels that there is no such thing as **the** perfect house; "The needs and desires of every client are so unique, so it follows that each should necessarily be unique."[9] It is likely that no amount of standard measures can identify one house that will suit everyone. This makes the design of houses an interesting problem to assist with a computer.

There have been many efforts to automate floor plan generation. Hahn *et al.*[8] demonstrate a method of generating building interiors in real-time as they are explored using a procedural algorithm that follows a set of simple rules. Bidarra *et al.*[2] use a hierarchical rule-based placement algorithm to create furnished living spaces with a variety of restrictions and heuristics. Bruls *et al.*[3] propose a visual representation for trees called "squarified treemaps" which Marson and Musse[10] use to generate balanced floor plans converting interior walls into hallways in order to ensure proper connectivity in the resulting house.

AI technologies have been applied to floor plan design. Mitchell *et al.*[12] use a brute force search for small instances of a layout problem ensuring desired room adjacencies. Martin[11] applies a multi-phase constructive procedural algorithm to quickly produce a batch of houses. Schnier and Gero[13] use a genetic

© Springer-Verlag Berlin Heidelberg 2011

program with a dynamic set of primitive functions in order to evolve designs where similarity to a given plan is the only requirement. Doulgerakis[5] uses a genetic programming algorithm. Activity assignment is accomplished by a procedural algorithm followed by the evaluation of the space considering many spatial, layout and functional requirements. He also considers polygonal spaces by allowing angled splits of rectangles.

This paper presents research that explores several methods and ideas with respect to automating the generation of floor plans[6]. Genetic algorithms are used to search for plans that satisfy user requirements. Multi-objective strategies are used to find a balance of the many objectives being optimized. Multiple answers provide a client with many options which satisfy the constraints in different ways. Final results may be hand-modified to fully satisfy the user's requirements, many of which may be difficult to formalize.

Our system has many possible applications. It is an interesting and challenging evolutionary design problem for evolutionary design algorithms. It can be used as a generator of inspirational ideas for architects and their clients. It can also be used to create dynamic environments for games, by generating new plans as needed, as well as automatically constructing very large expansive game environments. Lastly, it can be used for computer animations, where generating large environments for background scenery would otherwise be very time consuming.

This paper is organized as follows. Background information regarding evolutionary computation, evolutionary design and floor planning is found in Section 2. The system design is described in detail in Section 3. Section 4 shows the capabilities and flexibility of the system in a series of interesting problems. Section 5 sums up the effectiveness of this system in meeting the outlined goals, compares the system to some similar works and notes potential future work. Details of all this research are in the MSc thesis [6].

2 Background Information

2.1 Floor Planning

The requirements for houses can be somewhat informal. Many implicit requirements are derived from a combination of western culture and simple usability guidelines. They have been generally observed in most modern houses, and the ones that do not observe them usually give one that feeling as though something is out of place.

There are spatial requirements for a house. An implicit one is that the house footprint must be built to fit on the lot. As a result, room shapes and sizes must be designed accordingly to fit within the space. Rooms must be large enough to host their activities and furniture. There are also various layout requirements, as various rooms must be easily accessible or centrally located within the house. Lastly in order for a house to be functional, it must have certain rooms, such as a kitchen, bathroom, bedroom and a social room. In an autonomous system, these requirements either have to be implicit in the representation such that they are always satisfied, or explicitly measured as the quality of the solution.

Floor plan goals often vary from one culture to another, and one time period to another. As this paper is concerned with the constraints in the modern western society, the results and some of the goals may not coincide with other cultures. Nevertheless, these requirements could be tailored to other cultures as part of the problem specification. Many of these goals are flexible and highly subjective. However, if only the required constraints were satisfied the results would likely not be pleasing.

2.2 Evolutionary Design

A genetic algorithm [7] is a popular evolutionary algorithm which simulates a population of individuals evolving through natural selection. Evolutionary algorithms are often used in design problems [1]. They are well suited to problems where there is no clear solution, or explorative problems where there may be many acceptable solutions. Using a population-based stochastic heuristic search allows for a wide exploration combining a variety of good features from various solutions. Quality of solutions is either measured by a user interactively, or by a fitness function automatically. Additionally having a population of solutions allows the system to provide a multitude of reasonable answers to the problem.

2.3 Multi-Objective Analysis

In this paper there are many objectives to be optimized. This paper explores two popular strategies for combining the scores from multiple objectives into a single overall score or ranking: (i) Pareto Ranking; and (ii) Ranked sum.

Pareto ranking is commonly used in many multi-objective problem areas [7]. A solution is said to dominate another solution if the fitness in each of its objectives is at least as good as the other solution, and has a better score in at least one of the objectives. A rank of 0 is assigned to all non-dominated solutions. The next rank is assigned to the remaining undominated solutions. This is repeated until the population has been ranked.

Ranked sum is another means of multi-objective ranking [4]. The idea is to rank each objective separately within the population. The sum of these ranks is used to provide an overall rank for the individual. The sum of dominance ranks is used in this work. The dominance rank is the number of individuals which have a better value. This paper uses dominance ranks in the standard ranked sum. The advantage of using the ranked sum fitness evaluation is that unlike Pareto, outliers are not valued. This gives greater selective pressure towards solutions that are hopefully better overall. Normalized ranks may be used to favour more uniform improvement in all objectives. When computing a normalized rank, each objective rank is divided by the number of ranks for that objective.

3 System Design

The basic operation of the system is as follows. The genetic algorithm produces some chromosomes. Those chromosomes are converted to a physical floor plan

(Section 3.1). If the floor plan does not have room types, these are procedurally assigned(Section 3.2). The resulting floor plan is evaluated using the fitness evaluation(Section 3.3). The fitness scores are then given to the multi-objective evaluation scheme to generate single scores for use by the GA to select individuals for reproduction in the next generation.

3.1 Chromosome Representation

The system uses a grid similar to [12], whose size will be predetermined. The grid is fit to the bounding rectangle of the house exterior. The genotype representing a house configuration is shown in Fig. 1. Here, $h_1, h_2, ..., h_n$ correspond to the row heights, $w_1, w_2, ..., w_n$ correspond respectively to the column widths, and $x_{i,j}$ corresponds to the room type and cell number of the cell in row i, column j. The width and height of the grid rows and columns can be resized. Hence the representation is capable of describing any rectilinear house given a grid of a high enough dimensionality.

The transformation from genotype to phenotype proceeds by combining adjacent rooms of the same cell number into a single room, removing walls between those grid cells. The overall room type is given by a vote of the room types of all cells in each combined group. The final room type can be assigned using the procedural algorithm outlined in Section 3.2.

A modified form of the standard 2-point GA crossover is used. A rectangular selection is made by selecting two cell locations at random. Using this selection rectangle, the information from each parent is exchanged to create the children.

To allow creation of a building with several floors without modifying the chromosome or increasing the complexity in search space, a particular room within the floor plan is fixed in position, size and room type. In this way, any plan for the bottom floor can be combined with any plan for an upper floor to create the building.

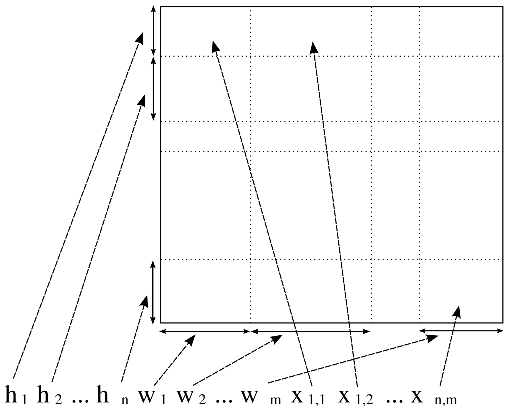


Fig. 1. Chromosome mapping to phenotype

Often it is necessary to design a house to fit a polygonal boundary. This may be due to spatial limitations or for aesthetic reasons. To construct non-rectangular houses, a floor plan is evolved to fit the bounding box of the outer footprint, which is then clipped to fit the shape. If the clipping removes an important room it will be penalized for that in the resulting fitness.

3.2 Procedural Activity Assignment

The procedural assignment used in this system is inspired by work in Martin [11] and Doulgerakis [5]. The following algorithm begins at the front door with one of a few allowable types, then assigns rooms in a breadth-first fashion.

```
proc assignType($room, types$) $\equiv$
  for $type := 1$ to $|types|$ step 1 do
      $types[type].value :=$ evaluateFeasibility($room, type$);
  od;
  sort($types, value$);
  for $type := 1$ to $|types|$ step 1 do
      $room.type = type$;
      for $adjroom \in$ adjacent($room$) $\cap$ unassigned($rooms$) do
          enqueue(($adjroom$, adjTypes($room$)))
      od;
      if $\neg$assignType(dequeue()) then $room.type = 0$; else return$true$; fi;
  od;
  return$false$;
```

3.3 Fitness Objectives and Evaluation

The requirement description for a house includes the type of a room (public or private), the minimum or maximum number of those rooms required, the minimum or maximum width, area and ratio of the room, the room types which should be close to the room, the room types which the room should be bigger than, and whether or not the room should have windows.

The calculation of an individual's fitness falls into several objectives relating to its adherence to the specifications in the requirements:

Functional: the building's adherence to the living requirements or required numbers of various room types.
Geometric: the buildings closeness to idealized geometric measurements.
Connectivity: how well the building satisfies certain proximities as specified by the requirements.
Reachable: how shallow and wide the graph of the building is. It measures the average number of rooms one must travel through from the entrance to reach any room in the building.
Ratio: how close the rooms are to their recommended ratio requirements.
Windows: how many rooms which require windows are not placed on the exterior of the house.

Table 1. Evolutionary Algorithm Parameters

Parameter	Value
# of runs	30
Population	500
Generations	200
Crossover	80%
Mutation	20%
Reset if stalled	10 generations
Selection Method	Tournament
Tournament Size	3
Assignment	Procedural
Diversity Factor	100
Ranking Method	Normalized Ranked Sum

3.4 GA Parameters

Evolutionary parameters are shown in Table 1. Most are standard in the literature [7]. A diversity preservation strategy is used. Duplicate individuals in the population incur a penalty to their rank of $diversity \cdot i$ where $diversity$ is the diversity preservation factor and i is how many times this individual has already been seen earlier in the population. Additionally, if the best individual has not changed for 10 generations, the population is reinitialized in order to evolve new layouts.

4 Results

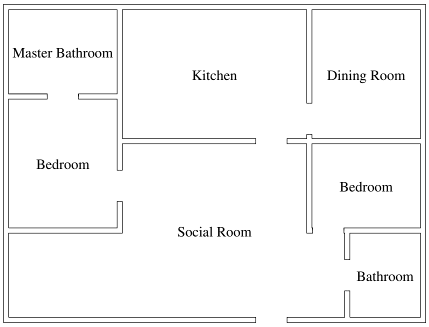
4.1 Analysis of Multi-Objective Algorithms

An experiment of a basic single floor 40'x30' house requiring two bedrooms was performed in order to evaluate various multi-objective strategies. As weighted sum is the traditional means of combining objectives, it is compared with the Pareto and ranked sum strategies.

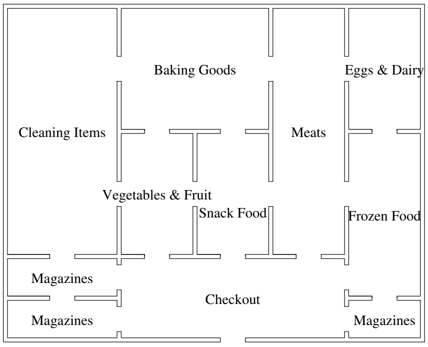
Tables 2 shows an analysis of the multi-objective strategies. In general, using normalized ranking produces objective values that are as good if not better than all other tested strategies. The second section shows the confidence of this strategy exceeding the fitness of the others on each objective. The result is that normalized ranked sum satisfies more requirements than other strategies. An example plan is shown in Fig. 2(a). It is similar to what might be seen in a contemporary condominium.

4.2 Polygonal Layout

Here, a half-hexagon shape is used to construct houses with an interesting exterior appeal. A 40' by 30' house is evolved and then clipped to this shape. The other requirements are fairly generic: 2 bedrooms with windows, kitchen with



(a) Basic house



(b) Grocery store

Fig. 2. Example Plans

Table 2. Average best of run fitness and confidence in Norm Ranked Sum outperforming other strategies

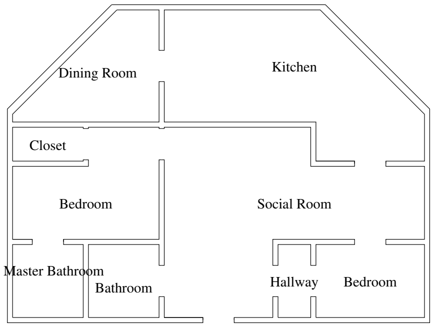
Objective	Connect.	Geom.	Funct.	Reach.	Ratio	Windows
Weighted	7.533	0.0	0.1333	1.789	0.2233	0.0
Pareto	7.0	72.06	0.0	1.904	1.063	0.0
Ranked	7.0	0.0	0.0	1.882	0.2016	0.0
Norm. Ranked	7.0	0.0	0.0	1.877	0.0	0.0
Weighted	99.9%	–	98.4%	–	94.9%	–
Pareto	–	99.5%	–	96.7%	99.9%	–
Ranked	–	–	–	88.9%	95.5%	–

adjoining living room, and others. Figure 3 shows several examples of the houses the resulted from this search. House (a) exceeds (b) in the geometric score as (b) has a small dining room. House (a) has an extraneous hallway from the social room leading to the bedroom. This is not directly punished, though there is less room for the remaining rooms in the house. House (b) has a large hallway with a corridor that does not lead anywhere on the bottom-left of the plan. This again is only punished indirectly in that the kitchen could have been larger. Otherwise the majority of requirements are successfully met in both plans.

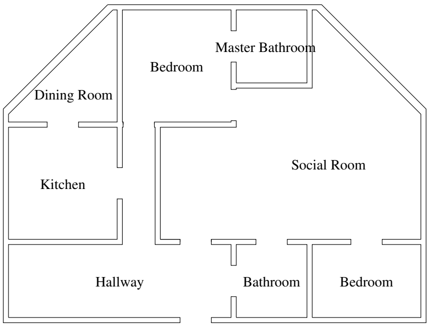
Figures 3(c) and 3(d) show what the plan from house (a) could look like furnished. The extra hallway was removed, which could have been done by a post-correction phase or evolved out with more time. There is a pleasing flow of traffic through the house, the plumbing is largely centralized and all of the rooms are of adequate size for their requirements.

4.3 2-Story Office Building

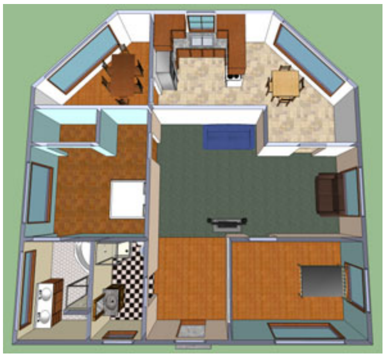
To show the flexibility of the system, another problem studied is that of an office floor plan. An office building needs to have offices and labs easily accessible via



(a)



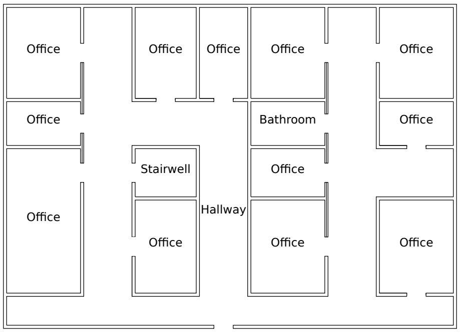
(b)



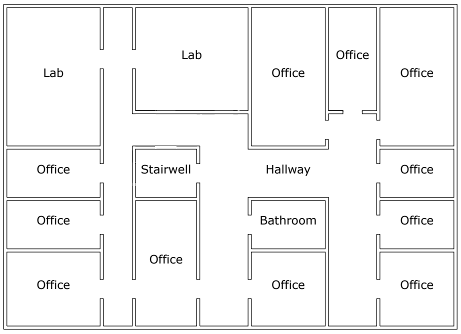
(c) 3D top down view of (a)

(d) 3D view of kitchen in (a)

Fig. 3. Half hexagon shaped houses evolved from system



(a) Bottom floor



(b) Upper floor

Fig. 4. Bottom and upper floor office building plans with a fixed stairwell

some system of hallways. Bathrooms need to be within easy reach of those offices. By affixing a stairwell in position, the system can separately evolve a bottom floor and an upper floor for the multiple story building. The evolved solutions succeed in meeting most of these requirements. The plans, shown in Fig. 4, required a few minor tweaks. The bottom floors had extra stairwells which were corrected to offices. The upper floors generated many hallways which have been combined into a single hallway.

4.4 Grocery Store

A final example is to evolve a layout that has discrete implicit regions even though they are not subdivided by walls. This is the case with a grocery store, where there are sections that provide various goods divided into differently sized regions. The customer is forced to walk through the checkout on leaving the store to ensure that they pay for their goods. The details of the requirements are available in [6]. Figure 2(b) shows an example plan evolved from the system.

5 Conclusion

This paper uses an evolutionary algorithms to solve a problem with a wide range of informal notions of optimality. New contributions include a hybrid representation, a multi-floor generation strategy, polygonal exteriors, and multi-objective strategies. The representation of the problem specifications allows it great flexibility in both problem depth and problem scope. The procedural assignment algorithm greatly reduces the search space of the problem by eliminating many useless layouts. This allows the algorithm to work on much larger spaces. Runs in this paper completed in approximately 7s on a single Intel Q6600 core. A variety of floor plans are presented to the user for further selection and hand-modification as desired.

Mitchell [12] and Doulgerakis [5] attempt to solve optimization problems in generating floor plans. Mitchell is concerned with ensuring the maximum number of desired adjacencies and afterwards adjusts sizes as a secondary goal, whereas this system attempts to optimize the two together. Doulgerakis uses a genetic program to construct houses through repeated angled divisions with a procedural activity assignment phase similar to ours. His objectives include adjacencies, room areas, ratios, and numbers of certain rooms, but does not include windows or distances through the house. The angled divisions seem to hurt the quality of layouts more. Other works [10,11] have not used search to optimize plans. None of the previous works allow for polygonal exteriors or multi-floor layouts.

There are many possible directions for this research. The chromosome could be modified to evolve multiple floors simultaneously. The sizes of the grid cells could be constant for all floors to ensure structural integrity as walls would be built over each other. Scaling up to even larger problems is challenging. This is a common problem in any search, which could be leveraged using modularization. An interactive/automatic hybrid evolutionary system could allow the user to interact with and influence the evolution. Parameters involved in procedural assignment could be evolved as part of the chromosome to help find ideal values for assigning room types. Since the true evaluation of a house is how well it serves the needs of its inhabitants, it may be possible to get a better rating of a house by running a simulation of agents using the house for various purposes.

Acknowledgements. This research supported by an NSERC PGS A awards and an NSERC Operating Grant 138467.

References

1. Bentley, P.J., Corne, D.W. (eds.): Creative Evolutionary Systems. Morgan Kaufmann Publishers Inc., San Francisco (2002)
2. Bidarra, R., Tutenel, T., Smelik, M.R.: Rule-based layout solving and its application to procedural interior generation. In: Proceedings of CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS) (2009)
3. Bruls, M., Huizing, K., van Wijk, J.: Squarified treemaps. In: Proc. TCVG 2000, pp. 33–42. IEEE Press, Los Alamitos (2000)
4. Corne, D.W., Knowles, J.D.: Techniques for highly multiobjective optimisation: some nondominated points are better than others. In: Proc. GECCO 2007, pp. 773–780. ACM, New York (2007)
5. Doulgerakis, A.: Genetic Programming + Unfolding Embryology in Automated Layout Planning. Master's thesis, University Of London (2007)
6. Flack, R.W.J.: Evolution of Architectural Floor Plans. Master's thesis, Brock University, St Catharines, Ontario, Canada (2011)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional, Reading (1989)
8. Hahn, E., Bose, P., Whitehead, A.: Persistent realtime building interior generation. In: Proc. Sandbox 2006, pp. 179–186. ACM, New York (2006)
9. Hirsch Jr., W.J.: Designing Your Perfect House. Dalsimer Press (2008)
10. Marson, F., Musse, S.R.: Automatic real-time generation of floor plans based on squarified treemaps algorithm. International Journal of Computer Games Technology (2010)
11. Martin, J.: Algorithmic beauty of buildings: Methods for procedural building generation. Tech. rep., Trinity University, San Antonio, TX, USA (2004)
12. Mitchell, W.J., Steadman, J.P., Liggett, R.S.: Synthesis and optimization of small rectangular floor plans. Environment and Planning B: Planning and Design 3(1), 37–70 (1976)
13. Schnier, T., Gero, J.S.: Learning genetic representations as alternative to hand-coded shape grammars. In: Proc. AI in Design 1996, pp. 39–57. Kluwer, Dordrecht (1996)

Path of Patches: Implementing an Evolutionary Soundscape Art Installation

José Fornari

Interdisciplinary Nucleus for Sound Communication (NICS),
University of Campinas (UNICAMP)
Campinas, São Paulo, Brazil
`tutifornari@gmail.com`

Abstract. Computational adaptive methods have already been used in Installation art. Among them, Generative artworks are those that value the artistic process, rather than its final product, which can now be of multimodal nature. Evolutionary Algorithms (EA) can be successfully applied to create a Generative art process that is self-similar yet always new. EAs allow the creation of dynamic complex systems from which identity can emerge. In computational sonic arts, this is ecological modeling; here named Evolutionary Soundscape. This article describes the development and application of an EA computing system developed to generate Evolutionary Soundscapes in a Multimodal Art Installation. Its physical structure uses paths of forking pipes attached to fans and microphones that capture audio to feed the EA system that creates the Soundscape. Here is described the EA system; its design in PureData (PD); its connection with the physical structure; its artistic endeavor and final sonic accomplishments.

Keywords: evolutionary computation, sonic art, soundscapes.

1 Introduction

In 2009, I presented a 20-hour workshop on adaptive methods for the creation of sonic art installations. I named it “aLive Music”, and was part of the package of my participations as advisor on the project of a music student who was awarded by the Brazilian art entity; MIS-SP (*Museu da Imagem e do Som, de São Paulo*) with a grant to develop his own multimodal project using PD (`www.puredata.info`). In this workshop, I presented an implementation in PD of an adaptive sound system, that uses principles of Evolutionary Algorithms (EA) for sound synthesis, which I developed in my Ph.D [8], [9]. Attending this workshop, was Fernando Visockis, a young artist, eager to learn adaptive computing methods to create Artwork Installations that could generate Soundscapes [4]. I was later invited to develop an Evolutionary Soundscape System that would be incorporated to an artwork installation, named “*Vereda de Remendos*” (VR); which is translated as “Path of Patches”. This artwork was awarded with “*Primeiras Obras*” prize, promoted by the cultural Brazilian entity CCJRC (“*Centro Cultural da Juventude – Ruth Cardoso*”). With this fund, the artists could build the installation. They assembled, tested, and finally exhibited it, from 14 to 29 of August, 2010, in Sao Paulo, Brazil.

© Springer-Verlag Berlin Heidelberg 2011

This work describes my contribution to the VR development, which involved the design and implementation of the Evolutionary Soundscape System to which the VR artwork physical structure was connected, and, therefore, capable of successfully generating a continuous synthesis of sound objects that constitute the Evolutionary Soundscape.

1.1 PA: Processual Artwork Defined

VR follows the Conceptual Art philosophy, which differs from most of the art developed during the history of Western culture. For centuries, the aim of any artist was to produce fine artistic objects, where the artist studied and developed techniques to build object of art, as best they could. The object was the final product of an artistic endeavor; and the reason and purpose for the existence of any artistic process. However, in the 1950s, probably due to the scientific arrival of digital electronics and computational science, artistic processes turned to be aesthetically noticed; slowly equating – and sometimes surpassing – the static materiality of the final object of art. This caused, in that decade, a breakthrough of new artistic possibilities and experimentations; most of them gathered by the term "Conceptual Art". To mention an example, Lucy Lippard, famous writer and art curator, when analyzing the artistic production of Sol LeWitt, in the 1960s, wrote that his work with structures was based on the premise that its "concept or idea is more important than the final object" [13]. This touches the definition of "Processual Artwork" (PA); a de-materialized expression of conceptual art that is complemented by the notion of Generative Art, here taking the definition of: any form of art where "a system with a set of defined rules and some degree of autonomy, is put on movement" [10]. Generative processes had been extensively explored in music and other forms of sonic arts; even before the arising of digital computing technology. More than four centuries ago, around 1650s, priest Athanasius Kircher, based on the belief that musical harmony should reflect natural proportions of physical laws, wrote a book entitled: *Musurgia Universalis*, where he described the design of a musical generating machine, based on these principles [3]. In 1793, Hummel published a system to generate musical scores, whose creation is attributed to Wolfgang Amadeus Mozart. This method generated music notation through a random process, based on the numbers given by tossing dices. This embeds most of late generative art elements, where a musician can create, from by applying simple rules, and building blocks (i.e. predefined musical bars), an astonishing amount of compositions. Later, this method was regarded as "Mozart's Dice Game" and has influenced many composers, such as John Cage and Hiller Lejaren, that created a musical piece entitled HPSCHD [11]. Generative art is therefore created by adaptive methods, which differ from deterministic and stochastic ones, as adaptive methods can rearrange their algorithmic structure according to the input.

1.2 From PAs to Soundscapes

PAs are seen as adaptive methods for art creation that uses generative processes where the artist is the element who provides the aesthetic judgement; constantly weighting the artistic value of the product, thus guiding the adaptive artistic process of art creation. In sonic arts, the correspondent of such process would be the design of sonic landscapes, also known as Soundscapes. This term was originally coined by Murray Schafer, that describes it as “natural, self-organized processes usually resultant of an immense quantity of sound sources, that might be correlated or not, but conveys a unique sonic experience that is at the same time recognizable and yet always original” [14]. This refers to the immersive sonic environment perceived by listeners that can recognize and even be part of it. Thus, a soundscape is also fruit of the listener's auditory perception. In this sense, a soundscape is categorized by cognitive units, such as: foreground, background, contour, rhythm, space, density, volume and silence. According to Schafer, soundscapes is sonically defined by 5 distinct categories of analytical auditory concepts, derived from their cognitive units. They are: Keynotes, Signals, Sound-marks, Sound Objects, and Sound Symbols. Keynotes are the resilient, omnipresent sonic aspects, usually unconsciously sensed by the listeners' perception. It refers to the musical concept of tonality or key. Signals are the foreground sounds that arouses listener's conscious attention, as they may convey meaningful information for the listener. Sound-marks are singular sounds only found in a specific soundscape, which makes this one perceptually unique. Sound objects – a term introduced by the french composer Pierre Schaeffer, extending the concept of musical note – can also be seen as a sonic process. Physical events create waveforms (a sound wave) carrying perceptual meaning, thus being heard, understood and recognized. Sound symbols are a more generic class that refer to sounds which evoke personal responses based on the sociocultural level of association and identity. In sonic arts, PAs are sometimes used to manipulate the process of creating these 5 units, to generate Soundscapes.

1.3 Soundscape and ESSynth

Evolutionary Sound Synthesis (ESSynth) was first introduced in [7], that describes its original method. A detailed explanation of ESSynth method is in [8, 9, 10]. As an overview, ESSynth is an Evolutionary Algorithm for sound synthesis based on manipulating waveforms, within a Population set. This was later implemented as an adaptive computational model in PD. Waveforms are seen as Sound Objects. They are the Individuals of the ESSynth model. Individuals have Sonic Genotypes, formed by acoustic descriptors – time-series expressing perceptually meaningful sonic aspects of sound objects, such as: Loudness, Pitch and Spectral density. ESSynth model has two sets of Individuals: Population and Target. Population set has the individuals that undergo the Evolutionary process. Target set has the individuals that do not evolve but influence the evolution process with their Genotypes. There are two dynamic processes operating in the Population set: Reproduction and Selection. Reproduction uses two Genetic Operators – Crossover (Recombination) and Mutation – to create new Individuals in the Population set, based on the information given by its Genotype. Selection process searches for the best Individual within the Population – the one whose Genotype is more similar with the ones in the Target set. This aims to emulate the natural pressure imposed by environmental conditions, that shape the evolutionary path of biological species. Selection uses a Fitness Function – a metric distance between Genotypes – to scan the Population set, measuring the Individuals' fitness distance, in comparison with the ones in the Target set. This process may also eliminate individuals not well-fit – the ones with fitness distance greater than a predetermined threshold – while selecting the best-one; the one presenting the smallest fitness distance. Reproduction uses the genetic operators: Crossover and Mutation, to create a new generation of individuals; the offsprings of the best-one, with each individual in the Population set. Initially ESSynth had a Population set with a fixed number of Individuals. There was a Generation cycle in which all a new generation of individuals in the Population entirely replaced their predecessor. The Reproduction occurred in Generation steps.

This new version of ESSynth replaced the fixed-size Population set by a variable-size one, with asynchronous Reproduction process, where the Individuals reproduce in pairs, and not in Generations steps. This also introduced the Lifespan rate, where Individuals now would be born, reproduce and die. With these developments, the sonic output of ESSynth was now created by all sound objects (the Individuals) co-existing in the Population set, instead of as before; where the sonic output was given by the single queue of best individuals. The most interesting aspect that arose from this new development was the fact that this new ESSynth model could now synthesize not only a single stream of sounds – as before – but a whole interaction of sonic events, co-existing as an Adaptive Complex System. This evolutionary process of sound objects seemed to endow Schafer's categories of analytical auditory concepts that define a Soundscape [17].

Soundscapes spontaneously occur in natural self-organized environments, but are very hard to be obtained “in vitro”. Formal mathematical methods and deterministic computational models are normally not able to describe the varying perceptual acoustic nature of a soundscape. In the same way, stochastic methods are normally not able in attaining sonic similarity. Deterministic and stochastic methods are mutually exclusive in describing soundscapes. As any mathematical or computational model, adaptive methods are also not complete, which means that they can not fully describe a phenomena.

In our case, the sonic process that generates soundscapes is the complex adaptive system, modeled by an evolutionary algorithm that is incomplete, as any other computational model, but is able to describe its adaptive process of reproducing and selecting sound objects; the case of ESSynth. The sound we hear from (and/or within) Soundscapes are, at the same time, acoustically new and yet perceptually similar. Several methods aiming to synthesize soundscapes were already been proposed, such as the ones described in [2]. Most of them are deterministic models that control the parameters of sound sources location and spacial positioning displacement. However, when compared to natural soundscapes, these methods don't seem capable of creating dynamic sonic processes that are self-organized and self-similar. From a systemic perspective, self-organization is a process that occurs in certain open complex systems, formed by parallel and distributed interaction of a variable number of agents (in our case; sound objects). The mind perceives the interaction between all internal and external agents as a system presenting emergence and coherence. As an example, the flocking behavior of biological population is a self-organized processes, as it is emergent, coherent and ostensive (i.e. it is perceptible). Soundscapes have, as well, similarity with gesture (i.e. intentional movement) as both rely on agents spatial location and self-organization to exist [15]. In this perspective, this new implementation of ESSynth fits well as a computer model that os able to generate a set of localized agents – a Population of spatially located sound objects – that reproduce new ones with genotypical features of their predecessors, which naturally creates self-similarity.

This article describes the usage of a new version of ESSynth – an evolutionary algorithm for sound synthesis – that creates the synthetic soundscape for VR; the artwork installation: ESSynth is an adaptive method that is able of emulating a sonic ecological system, in which sound objects are ephemeral beings. During their limited lifespan, they move inside a sonic location field – through the usage of sound location cues – and generate new sound objects that inherit sonic aspects of their predecessors. When a sound object lifespan is over, this is removed from the Population set; its genotype is erased from the genotype pool, to never be repeated again. The intrinsic property of ESSynth sonic output – the varying similarity – is paramount to generate synthetic soundscapes and thus being successfully used with artwork installations such VR; the case studied here. VR is an awarded multimodal project to which this customized version of ESSynth was designed. VR was also an ephemeral project per se. It was dismantled after its exhibition period was over. A video showing VR in action, with the evolutionary process of soundscape creation, is in the link: `http://www.youtube.com/watch?v=3pnFJswizBw`. This vanishing characteristic is common for processual multimodal artworks, since the first one, in the records: the *Poème Électronique*, created by Edgard Varèse, for the Philips Pavilion at the 1958 Brussels World's Fair; the first major World's Fair after World War II. Le Corbusier designed the Philips pavilion, assisted by the architect and composer Iannis Xenakis. The whole project was dismantled shortly after the fair was over, but its memory remains alive and still inspiring artistic creations to this date. A virtual tour to this installation was created by the "Virtual Electronic Poem" project, and can be accessed in the following link: (`http://www.edu.vrmmp.it/vep/`).

Section 2 describes the VR structural project; its concept, design and built. Section 3 explains the development of the customized version of ESSynth for VR, and its implementation in the open-source computing environment of PD (`www.puredata.info`). Section 4 describes the sonic results achieved by VR, as a multimodal processual artwork project, and discusses the conclusions found in this successful artistic project; commenting on an excerpt of an alive recording of the generated Soundscape and discussing their correspondence with natural Soundscapes, based on Schafer's categories of analytical auditory concepts.

2 A Plumbing Project

The bulging aspect of VR, seen in Figure 1 (left), is here understood as the dwelling of the evolutionary soundscape system, described in this article. Several segments of PVC plastic pipes – used in residential constructions – compose this plumbing structure. The artists built with this material several pathways that were connected at one vented end with fans, as seen in Figure 1 (right). These fans created a constant flow of air inside these interconnected and forked pathways, veering the air blowing, which created unusual sonic results. The pipes had four microphones inserted in specific parts of the plumbing system, to capture in real-time these subtle sonic changes.

The first concept of VR was inspired by a short story written by Jorge Luis Borges, entitled "*El jardín de senderos que se bifurcate*" (The Garden of Forking Paths), appeared in 1941, which is arguably claim to be the first hypertext novel, as it can be read – and thus understood – in multiple ways.

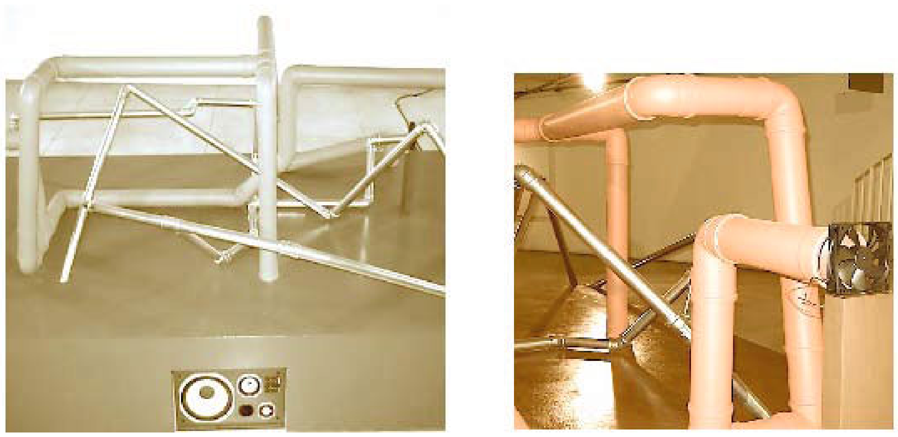


Fig. 1. Image of the VR installation. The plumbing structure (left), and a closeup of one fan attached to one of the vented ends of these pathways (right).

Contrary to the expected by the artists, this structure didn't generate the predicted sound, right after its assemblage. Actually, the system initially generate no sound at all. As in many projects, they had to go through a thorough tweaking of structural parts, fans positioning and microphones adjustments, in order to find the correct sweet-spots for the retrieval of veering venting sonic aspects, as initially intended. After the usual "period of despair", the sonic system finally came to live, working very alike the artists initial expectation, and kept on successfully generating the soundscape during all the period of its exhibition. When it was over, VR ought to be disassembled, by contract. Following the previously mentioned example of *Philips Pavilion*, VR was also dismantled. The artists, dressed as plumbers, took themselves the role of destroying it, bringing down this structure, as an artistic intervention. A video of the artistic disassembling of VR, in reverse mode, is in the following link: `http://www.youtube.com/watch?v=tQcTvfhcVb4&feature=player_embedded`

3 Evolutionary Synthesized Soundscapes

The dynamic generation of synthetic soundscape in VR needed a customized version of ESSynth. Although developed in 2003, during the Ph.D. dissertation of the author, the first implementation of ESSynth was finished in 2009, for the *RePartitura* artwork [6]. In this previous implementation – also programmed in PD – individuals where sound objects whose genotype were given by the mapping of graphic objects from handmade drawings. A new version of ESSynth was later developed to use as genotype other forms of artistic gesture, such as dance movements, as defined by Rudolf Laban [5]. In *RePartitura*, ESSynth didn't use a Selection process. It was enough to have only the Reproduction process. The soundscape was compounded by the unrestrained reproduction of individuals (sound objects) with short lifespan (10 to 30 seconds). This sufficed to create the variant similarity of a soundscape. This new ESSynth implementation for VR, a Selection process is back, using the same strategy described in the original ESSynth method from 2003. This Selection process basically measures the fitness distance between individuals. It calculates the Euclidean distance of the 6 arrays forming the sonic genotype, as seen in Figure 3. The author used PD for the implementation of such complex system because, besides the fact that PD is a free open-source software environment, multi-platform; it is also a robust computational platform for the implementation of real-time multimedia data processing algorithms, named – in PD terms – “patches”. PD also allows to explore meta-programming; which implies that the patch algorithmic structure can manipulate and create parts of itself, as well as other patches, which means that, it is possible to develop a patch that is adaptive and even evolutionary; that can reproduce other patches. The Reproduction process uses some of these strategies to simulate the dynamic creation of new individuals in the Population set; and in the individual lifespan inner process – to eliminate individuals and their genotypes from the genetic pool, when they reached their life expectation period, and die.

Figure 2 shows two structures programmed in PD, used in this implementation, depicted here to illustrate these algorithms. PD is a “data-flow”, or “visual” language. Patches are programmed using building blocks, called “objects”, interconnected so to describe the programming flow of events. PD was specifically developed for real-time processing. As such, it is possible to create structures in PD that are non-deterministic; allowing situations where two runs of the same patch, each time having the same inputs, may not have same results. This is interesting where simulating dynamic complex systems, such as the evolutionary algorithm for sound synthesis of ESSynth.

VR had four pipes pathways with fans attached at their ends. Initially, each pathway had inside one piezoelectric sensor to capture vibrations and the air blowing sound. Each sensor was connected to an audio channel, which summed up to 4 audio inputs for the ESSynth. Later they were replaced by microphones. The audio retrieved by each channel was filtered into three frequency regions: Low, Middle and High (as seen in Figure 2, left); which summed up to 12 frequency regions, for the 4 audio channels. Low region is under 100Hz (given by a first-order low-pass filter); Middle region, around 1000Hz (band-pass with Q=3) and High region, above 4000Hz (first-order high-pass). These 12 scalars referred to the 12 spectral regions of frequency, of the 4 input audio channels; captured by 4 microphones positioned inside the pipes. These data where inserted in the individuals sonic genotype, in the parameters arrays, denoted by the label “prmt-”, as seen in Figure 3.

Each Individual was designed as an instantiation of a PD *abstraction* (a coadjutant patch that works as a sub-patch and appears in the main patch as an object). Each active individual generated a sound object, created by computing models of non-linear sound syntheses. For this implementation, it was decided to use three models of sound synthesis: 1) Granular Synthesis (GS), 2) Karplus-Strong (KS), 3) Ring Modulation (RM). GS is a non-linear synthesis model that generates sound output

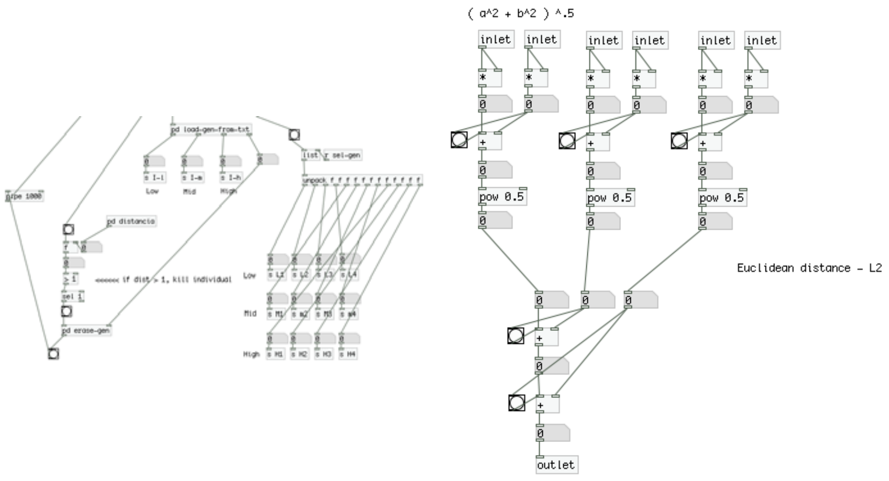


Fig. 2. Details of PD patches used in ESSynth. Part of the Selection process (left). Euclidean distance measurement of individuals genotypes (right).

from the overlapped looping of small waveforms, of about 1 to 100 ms; known as sonic "grains" [16]. KS is a physical model for the sound synthesis of strings. It was later developed in the Digital Waveguide Synthesis model [18]. It is constituted by a short burst of sound (e.g. white-noise pulse), a digital filter, and a delay line. The sound is recursively filtered in a feedback looping, which creates a sound output similar to a percussed string [12]. RM model heterodynes (multiply in the time domain) two waveforms; normally an input audio signal by another one generated by an oscillator. It is equivalent to the convolution of these audio signals in the frequency domain. This implies that the output sound will have the sum and difference of the partials of each input sound, thus this method generates new (and normally inharmonic) partials, sometimes delivering a bell-like sound.

The artists wanted the Soundscape to be generated by these synthesis models processing the audio from the commuting input of each one of the 4 audio channels. It was later decided to implement a logic to use the randomized commuting of one or more audio channels. When there was more than one input audio, they were mixed together, thus avoiding audio clipping. It may, at first, seems strange to use synthesis model to process sound input, instead of generating it. However, these 3 models are dependent of external audio data. GS uses sonic grains, which are provided by the segment of audio input. KS uses the same audio input to provide the pulse of audio for its feedback filtered looping. RM performs a frequency convolution with the audio input and a sine-wave.

The amount of processing over the audio input, for each synthesis model, was determined by 3 arrays of 100 elements each. These ones determine the processing rate of each synth effect; its intervening on the audio input, during the Individual lifespan. For each effect, an array of 100 elements was assigned. They are labeled as "intd-" which refer to the amount of each effect, along time. The elements of these arrays are normalized from [0,1]; 0 means no effect, and 1 means full effect. These arrays are part of the sonic genotype of each Individual, as seen in Figure 3.

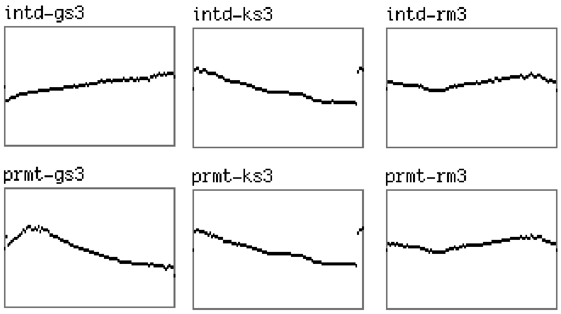


Fig. 3. Example of Sonic Genotype. It is formed by 6 arrays of 100 elements each. The arrays labels "intd-" refer to the synth effect along time. The arrays labeled "prmt-" refer to the synth parameters. Each synth effect has 2 arrays. Their labels are finished by the correspondent synth name: GS, KS and RM.

As seen in Figure 3, there are also 3 other arrays on the sonic genotype. They are labeled as "prmt-". These arrays refer to the parameters of each synthesis model. Although they are also arrays of 100 elements each, in this implementation, only the first 12 elements of each array were used. They are the control parameters related to the 12 frequency regions, as previously described. In future implementations, further parameters may turn to be necessary, therefore it seemed reasonable to leave these arrays with extra elements, even because it doesn't affect the computation required to run this system.

4 Sonic Results, Sound Conclusions

As any processual artwork, VR was, in several layers, a fleeting piece of art. Its structure was dismantled shortly after the exhibition was over. During its existence, the interactive parallel flow of sound objects, that constituted the soundscape, were constantly changing and vanishing. Each sound object, although perceptually similar, was never repeated. They were the product of an evolutionary process where individuals were coming into existence, reproducing and eventually dying, to never repeat themselves again, although passing to future generations their genetic legacy.

The achieved sonic result of VR was a synthetic soundscape that seems to follow Schafer's formal definition of natural soundscapes, as previously described. When listening to recordings of this installation, such as the one in the video, whose link is referred in page 4, it is possible to perceive some of Schafer's categories of analytical auditory concepts. Figure 4 shows the waveform (bottom) and the correspondent spectrogram (top) of an excerpt of this audio recording. The spectrogram depicts the partials conveyed in this waveform. The horizontal coordinate is time. In the waveform, the vertical coordinate is intensity, and in the spectrogram, it is frequency, where the colors changes refer to partials intensity (the darker the louder). Keynotes can be seen in this figure as formed by the structure of all darker horizontal lines, below 2KHz. They refer to the corresponding of a tonal center, normally associated to the cognitive sensation of resting. Each one of the horizontal lines refers to the concept of Signals. They have clear pitch, which are consciously perceived, and can be attributed to corresponding musical notes. Soundmarks are seen as the spread homogenous mass of partials, found along the spectrogram, which refers to the Soundscape identity.

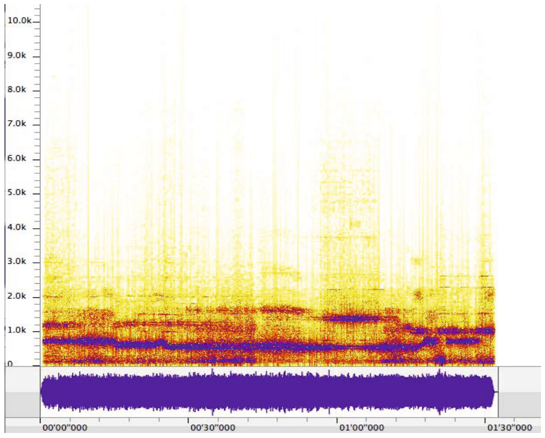


Fig. 4. Audio excerpt of a typical soundscape created by the system. The larger figure on top, shows its spectrogram. The narrow figure on the bottom is the waveform. Horizontal coordinate is time.

Acknowledgements

The author wants to thank the artists of "*Vereda de Remendos*", Fernando Visockis and Thiago Parizi – the PirarucuDuo (`http://pirarucuduo.tumblr.com`), that created, built and dismantled this multimodal installation, for the "*Centro Cultural da Juventude – Ruth Cardoso*", in São Paulo, Brazil; on August of 2010.

References

1. Blauert, J.: Spatial hearing: the psychophysics of human sound localization. MIT Press, Cambridge (1997)
2. Chowning, J.M.: The simulation of moving sound sources. In: Audio Engineering Society 39th Convention, New York, NY, USA (1970)
3. Cramer, F.: Words Made Flesh. In: Code Culture Imagination. Piet Zwart Institute, Rotterdam (2005); Truax, B.: Soundscape composition as global music: Electroacoustic music as soundscape*. Org. Sound 13(2), 103–109 (2008)

4. Fornari, J., Maia Jr., A., Manzolli, J.: Creating Soundscapes Using Evolutionary Spatial Control. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 517–526. Springer, Heidelberg (2007), doi:10.1007/978-3-540-71805-5.
5. Fornari, J., Manzolli, J., Shellard, M.: O mapeamento sinestésico do gesto artístico em objeto sonoro. Opus, Goiânia 15(1), 69–84 (2009); Edição impressa: ISSN 0103-7412. Versão online (2010), `http://www.anppom.com.br/opus` ISSN 1517-7017
6. Fornari, J., Shellard, M., Manzolli, J.: Creating Evolutionary Soundscapes with Gestural Data. SBCM - Simpósio Brasileiro de Computação Musical 2009. Recife. September 7-9 (2009)
7. Fornari, J., Shellard, M.: Breeding Patches, Evolving Soundscapes. Article. In: 3rd PureData International Convention - PDCon 2009, São Paulo, July 19-26 (2009)
8. Fornari, J., Maia Jr., A., Damiani, F., Manzolli, J.: The evolutionary sound synthesis method. In: International Multimedia Conference, Proceedings of the Ninth ACM International Conference on Multimedia. Ottawa, Canada. Session: Posters and Short Papers, vol. 9, pp. 585–587 (2001) ISBN:1-58113-394-4
9. Fornari, J.: Síntese Evolutiva de Segmentos Sonoros. Dissertação de Doutorado. Faculdade de Engenharia Elétrica e Computação – FEEC / UNICAMP. Orientador: Prof. Dr. Fúrio Damiani (2003)
10. Galenter, P.: What Is Generative Art? Complexity Theory as a Context for Art Theory in: Generative Art Proceedings, Milan (2003)
11. Husarik, S.: American Music, vol. 1(2), pp. 1–21. University of Illinois Press, Urbana (Summer 1983)
12. Karplus, K., Strong, A.: Digital Synthesis of Plucked String and Drum Timbres. Computer Music Journal 7(2), 43–55 (1983), doi:10.2307/3680062
13. Lippard, L.: Six Years. Studio Vista, London (1973)
14. Murray, R., Schafer, M.: The Soundscape (1977) ISBN 0-89281-455-1
15. Pulkki, V.: Virtual sound source positioning using vector base amplitude panning. Journal of the Audio Engineering Society 45, 456–466 (1997)
16. Roads, C.: Microsound. MIT Press, Cambridge (2001) ISBN: 0-262-18215-7
17. Shellard, M., Oliveira, L.F., Fornari, J., Manzolli, J.: Abduction and Meaning in Evolutionary Soundscapes. Book Chapter - pgs 407 - 428. Part of the Springer Book: Model-Based Reasoning in Science and Technology. Abduction, Logic, and Computational Discovery. Series: Studies in Computational Intelligence, Vol. 314. X, 654 p., Hardcover. ISBN: 978-3-642-15222-1. Springer, Heidelberg/Berlin. E-ISBN 978-3-642-15223-8 (2010)
18. `https://ccrma.stanford.edu/~jos/swgt/swgt.html`

Weighted Markov Chain Model for Musical Composer Identification

Maximos A. Kaliakatsos-Papakostas, Michael G. Epitropakis, and Michael N. Vrahatis

Computational Intelligence Laboratory (CI Lab), Department of Mathematics,
University of Patras, GR-26110 Patras, Greece
{maxk,mikeagn,vrahatis}@math.upatras.gr

Abstract. Several approaches based on the 'Markov chain model' have been proposed to tackle the composer identification task. In the paper at hand, we propose to capture phrasing structural information from inter onset and pitch intervals of pairs of consecutive notes in a musical piece, by incorporating this information into a weighted variation of a first order Markov chain model. Additionally, we propose an evolutionary procedure that automatically tunes the introduced weights and exploits the full potential of the proposed model for tackling the composer identification task between two composers. Initial experimental results on string quartets of Haydn, Mozart and Beethoven suggest that the proposed model performs well and can provide insights on the inter onset and pitch intervals on the considered musical collection.

1 Introduction

Various methods for computer aided musical analysis have been proposed, which focus on two different classes of data extraction models. Namely, the *global feature extraction models* [10], which encapsulate global features of a musical piece in a single value, and the *event models* which view the piece as a sequence of events that are described by their own value [6]. Global feature models have been used for musical genre identification [10,13] as well as for style and composer recognition [8]. Event models, have been utilized for chordal analysis [12], composer identification [4,5,7] and music composition [11] among others.

The composer identification task has been tackled through various event models, with the most successful approaches being with n-grams or $(n-1)$-th order *Markov chain models* [5,6,19]. These models utilize the Markov assumption that only the prior local context, i.e. some last few notes, may affect the next notes. A simple first order Markov chain model for composer identification has been recently proposed in [9]. This model provides information only for the sequence of scale degrees that form the melody, without incorporating any insights about their position in a phrase.

In the paper at hand, we propose to capture phrasing structural information from *inter onset* and *pitch intervals* of pairs of consecutive notes, and incorporate that information into the simple Markov chain model. To this end, assumptions can be made about which pairs of consecutive notes are of more or less musical importance. The proposed model is a weighted variant of the first order Markov chain model, namely the *Weighted Markov Chain* (WMC) model. Additionally, we propose an evolutionary

© Springer-Verlag Berlin Heidelberg 2011

procedure to automatically tune the two introduced parameters of the WMC model and consequently exploit its full potential for tackling the composer identification task between two composers. The performance of the proposed approach has been evaluated on string quartets of Haydn, Mozart and Beethoven with promising results.

The rest of the paper is organized as follows: Section 2 briefly describes how monophonic melodies can be expressed as Markov chains and how did that motivate us to propose the weighted variation of the Markov chain model, which is extensively described in Section 3. In Section 4, an automatic procedure for the application of the weighted Markov chain model on the musical composer identification task is presented. Section 5 presents experimental results of the proposed approach on a musical collection of string quartets of Haydn, Mozart and Beethoven. The paper concludes in Section 6 with a discussion and some pointers for future work.

2 Monophonic Melodies as Markov Chains

Monophonic melodies can be easily represented through a discrete Markov Chain (MC) model. A discrete MC is a random process that describes a sequence of events from a set of finite possible states, where each event depends only on previous events. Thus, each monophonic melody, as a sequence of discrete musical events (the scale degrees), can be formulated as a discrete MC model. Each MC is characterized by a *transition probability matrix* that represents the probability of transition from one state to an other. The aforementioned representation has been proposed in [9] to tackle a two way composer identification task between the string quartets of Haydn and Mozart.

To formulate each monophonic melody, and proceed with the construction of the probability transition matrix, we have to assume that any single note in the melody depends only on its previous note. This assumption has nothing to do with over-simplification of a musical composition, it is just a statistical compliance that lets us consider each monophonic melody as a first order MC. It is evident that, if one would like to utilize more information in each transition probability, a higher order MC should be incorporated [19], i.e. if each note is assumed to depend on its past two notes, we should incorporate a second order Markov chain.

Markov chain model intends to capture a composer's preferences on creating harmonic structures independently of the key signature of the piece, e.g. the $V \rightarrow I$ cadences. In [9], Markov chain models are built up for scale degree transitions in one octave, thus all musical pieces were transposed in the same key. To this end, a transition matrix of a musical piece M can be formally described by a square matrix T_M, where each pair (row, column) represents one possible state. Each element $T_M(i,j)$ of the transition matrix, describes the probability that scale degree class i is followed by scale degree class j.

Motivation: The Markov chain model as used so far, provides information only for the sequence of scale degrees that form the melody, regardless of the phrasing structure they are embodied. Two elements that could be possibly utilized to define phrasing structure are the *inter onset interval* and the *pitch interval* of a pair of consecutive notes that form each transition. The *inter onset interval* is the distance between two consecutive note start times, while *pitch interval* is the difference in semitones between two notes. A statistical refinement of the aforementioned two elements in a monophonic melody may lead to criteria about which transitions are of more or less musical importance.

For example, two consecutive notes with a great magnitude of their inter onset interval, possibly belong to a different phrase, thus their transition should not be considered of great importance. On the contrary, pairs of consecutive notes that are separated by small inter onset intervals may not constitute vital elements of the phrase they belong to, since they possibly form stylistic articulations. Similarly, a transition between notes that share pitch distance of great magnitude, e.g. over one octave, possibly reveals that these two notes belong to different phrases. The first one may belong to the end of the current phrase, while the second one to the beginning of the next phrase.

On the contrary, pairs of consecutive notes in a musical piece whose inter onset interval value is near to the mean value of all the inter onset intervals, probably constitute a pair of notes that demonstrates greater phrasing coherence. Thereby, they contain more information about the composer's musical character. The same holds for pitch intervals. It is evident that to capture the aforementioned characteristics, a weighting procedure on the transitions should be incorporated. This procedure should take into account both the inter onset and pitch interval of the notes.

3 Weighted Markov Chain Model

Motivated by the aforementioned comments, we try to capture qualitative statistical characteristics of musical nature and construct an enhanced transition probability matrix for each monophonic voice. The new transition probability matrix is constructed in a similar manner as the Markov chain model, with the difference that transitions are biased according to some musical criteria, that we will briefly discuss bellow. A general rule that can be deducted by the comments on Section 2, is that pairs of more distant time and pitch events provide less information than less distant ones. Thus an obvious question to face is: In which way that distance affects musical information? For example, do more distant notes contain linearly or exponentially less information than less distant ones?

An answer to the above questions could be provided by studying the distributions of the inter onset and pitch intervals of pairs of consecutive notes within a piece. Here, we study string quartets from classical composers such as, Haydn, Mozart and Beethoven. The string quartets of the classic music composers are perfectly suited for the composer identification problem since each string quartet is composed in four monophonic voices, where note transitions are clearly distinguished. A detailed description of the utilized musical piece collection along with their characteristics will be explained in the experimental results section (Section 5).

To this end, we have observed that all four voices from the string quartets of Haydn, Mozart and Beethoven follow a Gaussian-like inter onset and pitch interval distribution. Thus, we take as a test case a randomly chosen musical piece of each composer, to demonstrate the aforementioned behavior. Figure 1 illustrates normalized bar plots of pitch and inter onset intervals of the first violin for a string quartet of Haydn, Mozart and

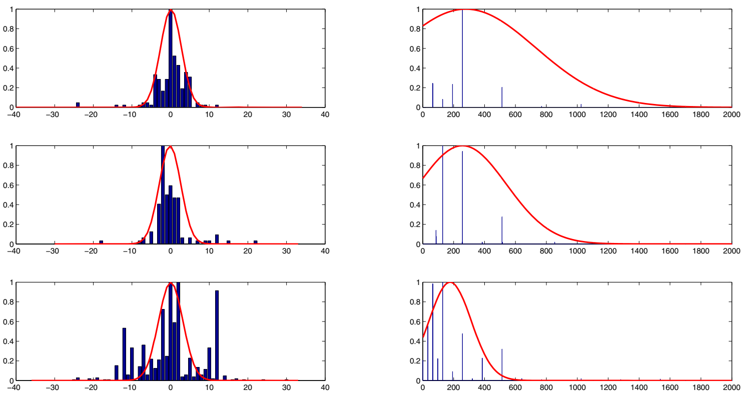


Fig. 1. Demonstration of the pitch (left) and inter onset intervals (right) along with their fitted Gaussian curves for the first violin of a musical piece of Haydn (top), Mozart (middle) and Beethoven (bottom)

Beethoven. The bar plot distribution of both pitch and inter onset intervals, exhibit that intervals follow a Gaussian-like distribution. This can be fairly justified by the Gaussian curves that are properly adjusted to fit the distributions.

It has been observed that all pieces in the considered collection, in all four violins demonstrate the same behavior. This observation provides insights on our question about how less important are more distant events. Since some intervals are preferred over others, they should contain more musical information. It would be reasonable to assume that pairs of consecutive notes that fall into an interval class that is more frequently used, should be given a greater weight than others. In other words, it could be assumed that there is a reciprocal relation on the usage frequency and the information contained in any inter onset or pitch interval. It is expected that each piece contains unique compositional information, thus different Gaussian distribution should be properly defined for the inter onset and pitch intervals of each piece.

Based on these observations it is rational to exploit the statistical characteristics of each piece and define a weighted variant of the Markov Chain (MC) model, that adjusts the weight of each transition within a piece, namely, the Weighted Markov Chain (WMC) model. More specifically, for a musical piece M we can define a Gaussian curve for the pitch intervals, $\mathrm{Pitch}_M(x, y)$, and the inter onset, $\mathrm{Time}_M(x, y)$, as follows:

$$\mathrm{Pitch}_M(x,y) = \exp\left(\frac{\left(p(x,y) - m_p(M)\right)^2}{2s_p(M)^2}\right), \tag{1}$$

$$\mathrm{Time}_M(x,y) = \exp\left(\frac{\left(t(x,y) - m_t(M)\right)^2}{2s_t(M)^2}\right), \tag{2}$$

where, x and y represent two consecutive notes of the current piece M that form a transition, $p(x, y)$ denotes the pitch interval, $t(x, y)$ denotes the inter onset interval,

$m_p(M)$ and $m_t(M)$ represent the mean values of all pitch and inter onset intervals respectively on the entire musical piece at hand, while $s_p(M)$ and $s_t(M)$ denote the standard deviation values of all pitch and inter onset intervals respectively.

Equations (1) and (2) capture the probabilistic characteristics that we have formerly discussed, and are capable to weight each transition $\mathrm{Trans}_M(x, y)$ from note x to note y, as the product of the multiplication of the inter onset and pitch intervals distribution values. More formally, $\mathrm{Trans}_M(x, y) = \mathrm{Time}_M(x, y)\,\mathrm{Pitch}_M(x, y)$. It is evident that, the distributions are different from piece to piece, since every piece has unique $m_t(M)$, $s_p(M)$ and $s_t(M)$ values. To keep the identical pitch transition $\mathrm{Pitch}_M(x, x)$ to a maximum weighted value 1, we consider from now on $m_p(M) = 0$ for all musical pieces. In Equation (1) a transition from pitch x to pitch x has a value $p(x, x) = 0$, to ensure that $\mathrm{Pitch}_M(x, x) = 1$, we should set $m_p(M) = 0$ for every musical piece. One can observe that some values of the distributions vanish to zero, even in cases where exist some transitions worth mentioning. For example, on the bottom left plot of Fig. 1, we observe that transitions with pitch interval value $p(x, y) = 12$, exhibit a high value bar plot and should not be ignored. Although the value of the fitted Gaussian curve tends to vanish that transition.

To overcome the aforementioned problem, we introduce two parameters, r_1 and r_2, to properly adjust the *stiffness* of the Gaussian-like curves, described by Eqs. (1)-(2). The adjusted distribution values can be defined according to the following equations:

$$\mathrm{Pitch}_M^{r_1}(x, y) = \exp\left(r_1 \frac{\left(p(x, y) - m_p(M)\right)^2}{2 s_p(M)^2} \right), \tag{3}$$

$$\mathrm{Time}_M^{r_2}(x, y) = \exp\left(r_2 \frac{\left(t(x, y) - m_t(M)\right)^2}{2 s_t(M)^2} \right), \tag{4}$$

where $r_1, r_2 \geqslant 0$. In the extreme case where $r_1 = r_2 = 0$ all transitions will have the same weight value, equal to 1, which is the simple Markov chain model. In cases where both parameters are $r_1, r_2 > 1$ the fitted distribution exhibits a tight "bell"-shape around the mean value, while in the opposite case a less tight one.

To demonstrate this behavior, we exhibit in Fig. 2 the effect of the r_1 and r_2 variables on pitch and inter onset interval curves, for three different values ($0.1, 1$, and 10 respectively). Finally, we define and use from now on, as weight of the transition from note x to note y the product of the aforementioned Eqs. (3)-(4).

$$\mathrm{Trans}_{(r_1, r_2)}(x, y) = \mathrm{Pitch}_M^{r_1}(x, y)\,\mathrm{Time}_M^{r_2}(x, y). \tag{5}$$

The construction of the new transition matrix follows the same procedure as in the Markov model, with the only exception that not all transitions are equally weighted, but given a set of $r1, r2$ parameters, the (x, y) element of the matrix will be weighted by Eq. (5).

4 Weighted Markov Chain Model for Composer Identification

In this section, we incorporate the *Weighted Markov Chain* (WMC) model into a general procedure to effectively tackle the musical composer identification task between

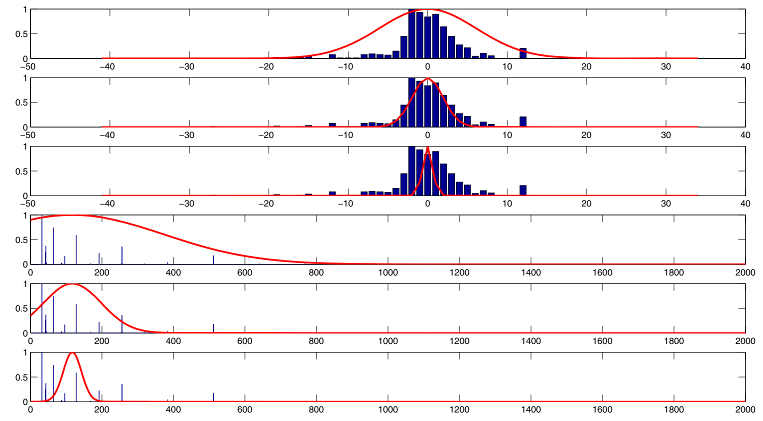


Fig. 2. Adjusted Gaussian curves over the pitch intervals distribution (top three), with $r_1 = 0.1$, $r_1 = 1$ and $r_1 = 10$ respectively, and over the inter onset intervals distributions (bottom three), with $r_2 = 0.1$, $r_2 = 1$ and $r_2 = 10$ respectively

two composers. The WMC model is a properly defined model for the representation of a musical piece, since it captures some essential characteristics for both pitch and inter onset intervals. The WMC model utilizes two parameters for adjusting the Gaussian curves that estimate the distributions of pitch and inter onset intervals of a given piece. These parameters should be automatically tuned to properly capture the characteristics of the composer style. Thereby, we propose the incorporation of any intelligent optimization algorithm to efficiently tune WMC model parameters and consequently tackle the musical composer identification task by maximizing the classification performance, i.e. the identification accuracy, of the proposed methodology.

Suppose that we have a collection of monophonic melodies of musical pieces composed by two composers. In order to achieve maximum composer identification accuracy with the WMC model, we perform the following five steps:

1. Provide an intelligent optimization algorithm to properly adjust r_1 and r_2 parameters.
2. Given r_1 and r_2 values, transform the monophonic melody of each piece in the collection into a weighted transition probability matrices through the WMC model.
3. Classify the weighted transition probability matrices by performing a leave one out classification task, using any supervised classifier.
4. Calculate the success percentage of the classification task and use it as a fitness value of the optimization algorithm for the r_1 and r_2 parameters.
5. Repeat the above procedure until either a termination criterion is reached or the r_1 and r_2 values produce the best composer identification accuracy for the utilized classifier.

More specifically, in this work, we propose to utilize as a supervised classifier, the well known Support Vector Machine (SVM) classifier [3,16], which has been successfully applied to a wide range of difficult classification problems [16]. SVM is a supervised binary classifier which is trained with a set of paradigms. It classifies a paradigm to one of the two categories by representing it as a point in the classification space. It is mapped in a way that all paradigms of the two categories can be separated by a clear gap which is as wide as possible. When a new paradigm is presented, SVM classifies it based on which side of the aforementioned gap, i.e. respective class, will fall into.

Additionally, to optimize the r_1 and r_2 parameters, we employ a global optimization algorithm that can handle nonlinear, (possibly) non-differentiable, multi-modal functions, namely the Differential Evolution (DE) [15,17]. DE is a population-based stochastic algorithm, which utilizes concepts borrowed from the broad class of the Evolutionary Algorithms. It exploits a population of potential solutions to effectively explore the search space and evolve them by simulating some basic operations involved in the evolution of genetic material of organism populations, such as natural selection, crossover and mutation. A thorough description of DE can be found in [15,17].

5 Experimental Results and Concluding Remarks

In this section, we perform an experimental evaluation of the proposed methodology over a musical piece collection formed by string quartet movements of Haydn, Mozart and Beethoven. The compilation contains 150 movements of string quartets in MIDI format from collections [2,14], 50 pieces for each composer. All movements were composed in a major scale and transposed in the key of C major.

It has to be noticed that, string quartets have a strictly specified structure, which makes them a difficult musical collection for composer identification. Every string quartet is composed in four voices, each of which play a certain role in the composition. The voice of the higher pitch register plays the leading voice role, the others form the harmony and the lower voice also forms almost always the bass line. The four voices are almost always monophonic and can be easily separated in four different monophonic tracks. When polyphony occurs, the skyline algorithm is performed to keep only the higher note of the polyphonic cluster [18]. For the rest of the paper, we refer to the highest voice, as the first voice, the next lower as second and similarly for the third and fourth.

More technically, in this work we utilize for the optimization task the DE/rand/1/bin strategy, with a population of 20 individuals, and default values for the parameters as stated in [17] i.e. $F = 0.5, CR = 0.9$. The number of maximum generations is kept fixed and equal to 200. Additionally, we utilize as a classification method the SVM with default parameters as stated in the libSVM library [1]. It has to be noted that the transition matrix of each voice is reshaped into a normalized vector to meet the SVM formality. For each of the 150 movements, we have four separate monophonic voices. Thus, we can split the experimental procedure and perform four different simulations to study and observe the composers unique compositional style of every voice.

Table 1 exhibits experimental results of the Markov Chain (MC) model and the Weighted Markov Chain (WMC) model for all four voices of three different composer identification tasks, namely Haydn–Mozart, Beethoven–Haydn and Mozart–Beethoven. The following notation is used in Table 1: *Voice* indicates which voice we refer to; *MC–success performance* indicates the classification success performance of the simple Markov Chain model; similarly *WMC–success performance* indicates the mean value of the classification success performance over 30 independent simulations produced by the

Table 1. Experimental results for the simple Markov Chain (MC) model, the Weighted Markov Chain (WMC) model and the corresponding r_1 and r_2 mean values

Haydn – Mozart					
Voice	MC Success Performance	WMC Success Performance	Improvement	r_1	r_2
First	63%	65%	2%	0.6760	15.1925
Second	53%	70%	17%	0.8025	5.7154
Third	57%	67%	10%	2.7398	18.1810
Fourth	55%	63%	8%	3.0347	9.9962
Beethoven – Haydn					
Voice	MC Success Performance	WMC Success Performance	Improvement	r_1	r_2
First	66%	88%	22%	3.1389	0.1204
Second	71%	75%	4%	1.3641	0.8872
Third	61%	59%	-2%	3.5439	2.3682
Fourth	64%	78%	14%	0.0071	8.9264
Mozart – Beethoven					
Voice	MC Success Performance	WMC Success Performance	Improvement	r_1	r_2
First	82%	87%	5%	6.5461	0.9697
Second	68%	74%	6%	3.9459	0.4409
Third	67%	71%	4%	0.1840	2.9747
Fourth	70%	77%	7%	0.7016	5.1395

weighted Markov chain model; *Improvement*, denotes the improvement percentage of the WMC versus the simple MC model. Finally, for the aforementioned WMC–success performance results, we exhibit the mean best values of the r_1 and r_2 parameters.

A first comment is that, in the majority of the considered cases, the WMC model improves the simple MC model, even by a small amount. In the third voice of the Beethoven–Haydn identification task, DE has been trapped in a local maximum, since when $r_1 = r_2 = 0$ the identification success would be the same as the simple MC model. The WMC model exhibits a good overall improvement in the Beethoven–Haydn and Haydn–Mozart tasks. In the data sets of Mozart–Beethoven and Beethoven–Haydn, we observe that in the first two voices $r_2 < 1$, while $r_1 > 1$. The opposite happens for the third and the fourth voice, with exception of the r_1 factor in the third voice of the Beethoven–Haydn set. This might indicate that Beethoven's aspect for the operation of the four voices in the string quartets, is different compared to Haydn's and Mozart's. Thereby, Beethoven seems to utilize smaller pitch intervals within phrases for the first two voices, and smaller inter onset internals within phrases for the third and the fourth voice.

Next we proceed with some comments of musical nature that we can make about the Haydn–Mozart pair which is one of the most difficult identification task considered in this work. In this task, we observe that the smallest improvement was made for the first voice. This could be due to the fact that the first voice already contains most of the information, since it plays the role of the leading voice. It also explains the fact that in this voice the simple Markov chain model produced its best success performance over the utilized collection. Additionally, the r_1 values were kept at a low level near to zero, which possibly means that distant pitch intervals contain considerable information. Musically, this signifies that notes with distant pitches may belong to same phrases. A similar behavior also holds for the second voice. On the contrary, in the first voice, values of the r_2 parameter were kept in high levels, near 15, which possibly means that distant time events are less important, resulting that distant time events are used as phrase separators. A similar behavior can also be observed for the third voice. Finally, the third and the fourth voices exhibit best results for high values of the r_1 parameter. Thus, it could be said that different use of closely neighboring pitches is unique for the string quartets of Haydn and Mozart.

Similar comments can be made for the remaining identification tasks, by analyzing pairs of the r_1 and r_2 parameter values of different voices. A further analysis should be made by musicologists, who could extract refined information by the aforementioned comments, in the considered musical collection. To collect more musical structural meanings, we can enhance the proposed model by incorporating more variables that could probably capture additional characteristics of the musical collection.

6 Conclusions

In this work, a weighted version of the simple Markov chain model for the representation of a monophonic musical piece, which includes further information about its pitch and inter onset intervals has been proposed. Furthermore, to tackle the composer identification task an automatic procedure based on an evolutionary algorithm for tuning the parameters of proposed model has been presented. Initial experimental results suggest that it is a promising model. The proposed model has two main aspects. Firstly, as the results suggest, the weighted Markov chain model performs better than the simple Markov chain model and in some cases exhibit high classification performance. Secondly, the incorporation of the proposed weighted Markov chain model representation can provide insights about the behavior and the characteristics of the pitch and inter onset intervals on each monophonic voice in a given musical collection. Thereby, models like the proposed one, in accordance with the essential opinion of a musicologist, could lead to significant insights for identification of classical composers.

Several interesting aspects should be considered in a future work. The proposed approach should be further studied in a bigger collection with either string quartets or pieces that share similar voice structures. The combination of all four voices into a single transition matrix may further enhance the performance and the musical quality of the proposed approach. Additionally, the proposed approach could be also tested for sub-genre classification, since the compositions within a genre, often utilize certain combinations of musical instruments and very similar voice structures. Furthermore, genre classification could be performed, with a proper matching of the roles of monophonic voices between different genres.

References

1. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

2. Classical MIDI Connection: The classical MIDI connection site map, `http://www.classicalmidiconnection.com`
3. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20, 273–297 (1995)
4. Geertzen, J., van Zaanen, M.: Composer classification using grammatical inference. In: Ramirez, R., Conklin, D., Anagnostopoulou, C. (eds.) Proceedings of the International Workshop on Machine Learning and Music (MML 2008), Helsinki, Finland, pp. 17–18 (July 2008)
5. Hillewaere, R., Manderick, B., Coklin, D.: Melodic models for polyphonic music classification. In: Proceedings of the 2nd International Workshop on Machine Learning and Music (MML 2009), Bled, Slovenia, September 7, pp. 19–24 (2009)
6. Hillewaere, R.: M, B., Conklin, D.: Global feature versus event models for folk song classification. In: Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009), Kobe, Japan, October 26-30, pp. 729–733 (2009)
7. Kaliakatsos-Papakostas, M.A., Epitropakis, M.G., Vrahatis, M.N.: Musical composer identification through probabilistic and feedforward neural networks. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 411–420. Springer, Heidelberg (2010)
8. Kranenburg, P.V., Backer, E.: Musical style recognition - a quantitative approach. In: Parncutt, R., Kessler, A., Zimmer, F. (eds.) Proceedings of the Conference on Interdisciplinary Musicology (CIM 2004), Graz, Austria, April 15-18, pp. 1–10 (2004)
9. Liu, Y.W.: Modeling music as markov chains: Composer identification (June 2002), `https://www-ccrma.stanford.edu/~jacobliu/254report.pdf`
10. Mckay, C., Fujinaga, I.: Automatic genre classification using large high-level musical feature sets. In: Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR 2004), Barcelona, Spain, October 10-14, pp. 525–530 (2004)
11. Orchard, B., Yang, C., Ali, M., Verbeurgt, K., Dinolfo, M., Fayer, M.: Extracting patterns in music for composition via markov chains. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 1123–1132. Springer, Heidelberg (2004)
12. Pardo, B., Birmingham, W.P.: Algorithms for chordal analysis. Computer Music Journal 26(2), 27–49 (2002)
13. Ponce De León, P.J., Iñesta, J.M.: Statistical description models for melody analysis and characterization. In: Proceedings of the 2004 International Computer Music Conference (ICMC 2004), pp. 149–156. International Computer Music Association, University of Miami, Miami, USA (2004)
14. Prévot, D.: T16 string quartets, `http://www.lvbeethoven.com/Oeuvres/Music-Midi-Mp3-String-Quartets.html`
15. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series. Springer-Verlag New York, Inc., Secaucus (2005)
16. Steinwart, I., Christmann, A.: Support Vector Machines. In: Information Science and Statistics, 1st edn. Springer, New York (2008)
17. Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
18. Uitdenbogerd, A.L., Zobel, J.: Manipulation of music for melody matching. In: Proceedings of the sixth ACM International Conference on Multimedia - MULTIMEDIA 1998, Bristol, United Kingdom, pp. 235–240 (1998)
19. Wolkowicz, J., Kulka, Z., Keselj, V.: n-gram based approach to composer recognition. Archives of Acoustics 33(1), 43–55 (2008)

SANTIAGO - A Real-Time Biological Neural Network Environment for Generative Music Creation

Hernán Kerlleñevich, Pablo Ernesto Riera, and Manuel Camilo Eguia

Laboratorio de Acústica y Percepción Sonora, Universidad Nacional de Quilmes
R. S. Peña 352, Bernal, 1876, Buenos Aires, Argentina
{hk,pr,me}@lapso.org
http://lapso.org

Abstract. In this work we present a novel approach for interactive music generation based on the dynamics of biological neural networks. We develop *SANTIAGO*, a real-time environment built in Pd-Gem, which allows to assemble networks of realistic neuron models and map the activity of individual neurons to sound events (notes) and to modulations of the sound event parameters (duration, pitch, intensity, spectral content). The rich behavior exhibited by this type of networks gives rise to complex rhythmic patterns, melodies and textures that are neither too random nor too uniform, and that can be modified by the user in an interactive way.

Keywords: generative music, biological neural networks, real-time processing.

1 Introduction

Complex systems offer an outstanding opportunity to work in generative music. One important message that came out from the study of complex systems is that very simple rules can lead to complex behavior, starting both from an ordered or from a completely disordered state. In this way, complex systems emerge as a new state that supersedes the dichotomy between order and disorder, being regular and predictable in the short term but unpredictable in the long run [6].

It is interesting to trace an analogy with what happens in generative music, where the key element is the system to which the artist cedes partial or total subsequent control [5]. Incorporating complex systems into generative music potentially allows the emergence of a variety of stable and unstable time structures, that can create expectancies in the listener and also deceive them, in a similar way to what happens with tension-distension curves in composed music [8].

Among complex systems, biological neural networks are probably the ones that exhibit the most rich behavior. In fact, the brain is the most complex device that we can think of. In its simplest realization a neural network consists of units (neurons) connected by unidirectional links (synapses). The units are

© Springer-Verlag Berlin Heidelberg 2011

characterized by its (voltage) level that can be continuous or discrete. We distinguish between artificial neural networks, where units are highly simplified models and the complexity resides in the network, and biological (or biophysical) neural networks where the model neuron is fitted to adjust the records obtained in real biological neurons and has a rich intrinsic dynamics.

There are some previous works that incorporate the behavior of neural networks into rules for generative music [3]. The most disseminated tool has been artificial neural networks, that have been applied to music in many ways, ranging from pitch detection [17], musical style analysis [16] and melody training [4] to composition purposes [14]. Most of these works were based on the idea that complex behavior emerges from the network only, disregarding the intrinsic dynamics of each neuron unit. In fact, in artificial neural networks the units are modeled with a single continuous or two-state variable.

However recent studies on the behavior of neural networks have shown that the intrinsic dynamics of the neurons plays a major role in the overall behavior of the system [11] and the connectivity of the network [2]. In this work we take into account the complexity emerging from the intrinsic dynamics of the units and from the connectivity of network as well. In order to do this, we adopt biologically inspired models for the neurons, that produce spikes, which are considered the unit of information in the neural systems. In simple terms, a spike is a short-lasting event in which the voltage of the neuron rapidly rises and falls. A review of some biological neuron models can be found in [10].

Also, previous works treated the networks as input/output systems, biased with the task-oriented view of artificial neural networks. In contrast, our work includes all the events (spikes) from the network as relevant information, as small neuron networks are supposed to do. Among the many realizations of small neural networks in the animal realm, we take those that are able to generate rhythmic behavior, with central pattern generators (CPGs) as a prominent example. CPGs are small but highly sophisticated neural networks virtually present in any animal endowed with a neural system, and are responsible for generating rhythmic behavior in locomotion, synchronized movement of limbs, breathing, peristaltic movements in the digestive tract, among others functions [7]. In this CPG view, the individual spikes can be assigned to events in our musical generative environment (such as notes) that can lead to complex rhythmic behavior.

In addition to rhythm generation, spike events can trigger complex responses in neural systems [13], and modulate dynamical behavior. In this spirit, we also use biological neural networks to modify the pitch, spectral content, duration and intensity of the generated events.

Finally, we adopt excitatory-inhibitory (EI) networks as the paradigmatic network. EI networks arise in many regions throughout the brain and display the most complex behavior. Neurons can be either excitatory or inhibitory. When a spike is produced in an excitatory (inhibitory) neuron, this facilitates (inhibits) the production of spikes in the neurons that recieve the signal of that neuron through a synapse.

The realization of the ideas expressed above is *SANTIAGO*, a real-time biological neural network environment for interactive music generation. It is named after Santiago Ramón y Cajal (1852-1934) the spanish Nobel laureate, histologist and physiologist, considered by many to be the father of modern neuroscience.

The article is organized as follows. In section 2 we expose the concept and design of *SANTIAGO*. In section 3 we show some results. In section 4 we present our conclusions.

2 *SANTIAGO* Description

2.1 Overview

Starting with a simple yet biologically relevant spiking neuron model, our environment allows to build up neural networks that generates a musical stream, creating events and controlling the different parameters of the stream via the action of individual spikes or some time average of them (firing rate).

SANTIAGO is designed with a highly flexible and modular structure. It is possible to build small networks that only create rhythmic patterns, where each neuron generates events (spikes) of a certain pitch, intensity and timbre (see the example in Section 3), and progressively add more neurons or other small networks that control the pitch, intensity or spectral content of the events generated by the first network. Also, different temporal scales can be used in different sub-networks, thus allowing the spike events to control from long term structures to granular synthesis.

In fig. 1 we display a diagram of a very basic realization of our environment. Excitatory neurons project excitatory synapses (black dots) to other neurons

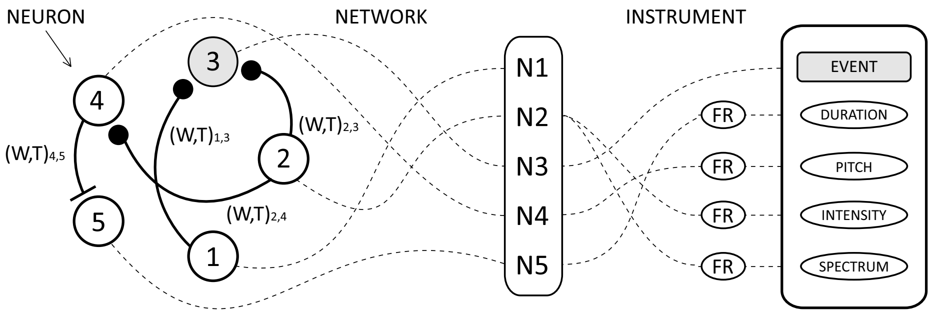


Fig. 1. Diagram of a simple realization of *SANTIAGO*. Five neurons are connected forming a small EI network (left). Neuron 1 excites neuron 3, neuron 2 excites neuron 3 and 4, and neuron 4 inhibits neuron 5. Each synapse is characterized by its synaptic weight W and time delay T. Instrument terminals N1-N5 map the output of each neuron (dotted lines) to events or event parameters (right). The notes (events) of this instrument will occur when N3 fires a spike, the pitch of the note will be determined by the averaged firing rate (FR) of N4, the durations of the notes by the FR of N5, and the intensity and the spectral content will be controlled by the FR of N2.

and inhibitory neurons project inhibitory synapses (bars) to other neurons. Each synapse is characterized by its weight W (how strong is its action) and some time delay T, which in real synapses corresponds to the physical length of the link (axon). The EI network will have a complex behavior as a result of the intrinsic dynamics of the neurons, which depends on its internal parameters, and the connectivity of the network (connectivity matrix). The user can control all these parameters in real time in order to obtain a desired result or explore a wide diversity of behaviors.

The outputs of this network are the spikes generated by all the neurons (N1-N5). These events are injected into the Instrument which has the main responsible to map the spike events to sound.

As we mentioned in the previous section, neurons of the network can act as event generators or modulation signals. In the first case, every spike triggers an event or note. In the second case an averaged quantity, the firing rate (number of spikes in a fixed time window), is calculated and mapped to a certain sound event parameter.

2.2 Implementation

SANTIAGO is basically built as a set of abstractions for Pd-Extended (version 0.41.1) [15]. It is completely modular and works under different platforms. It also uses the Gem external for neural activity visualization[1].

The current version implements the neuronal model with the *fexpr~* object [18], which permits access to individual signal samples, and by entering a difference equation as part of the creation argument of the object the output signal gives the integrated dynamical system. The environment runs flawlessly with 10 neurons, using 60% of CPU load on a Intel Core2Duo 2.2 Ghz machine. In the present we are implementing the model with external objects developed specially for this application that will increase the number of possible units.

OSC communication has been implemented with *mrpeach* objects, enabling live music creation across local networks and the internet. On the other hand, the network capability can be used for parallel distributed computing, when the number of neurons is high or the desired output demands too much load for a single machine. For this purpose, when the main patch is loaded, the user can establish a personal Machine ID number, which will be sent as a signature for every message and event generated from that computer. Finally, all streamed data is easily identifiable and capable of being routed and mapped dynamically.

In this version, the inputs of the environment are: keyboard, mouse and MIDI controllers. The outputs include real-time MIDI controllers and OSC messages, multichannel audio and visuals.

A single-neuron version of *SANTIAGO* (SOLO) was also created, for the user to explore the possibilities of one neuron only, and its different firing modes.

2.3 Architecture

The implementation of *SANTIAGO* is still in development. Yet it is very functional even for complex interactions. The environment is built upon a modular structure, which is described next. The core of the system consists of three main structures: neurons, network and instruments. Each structure has a window, accessible from the Main Panel.

Neurons. This structure allows the creation of *neuron modules*, which constitute the minimal units of the system. The implemented neuron model was originally proposed by Izhikevich[9], as a canonical model capable of display a large variety of behaviors. It is described by a system of two differential equations (Eqs. 1a and 1b) and one resetting rule (Eq 1c)

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I(t) \tag{1a}$$

$$\frac{du}{dt} = a(bv - u) \tag{1b}$$

$$if v \geq 30\,mV,\ then \begin{cases} v \longleftarrow c \\ u \longleftarrow u + d \end{cases} \tag{1c}$$

The dynamical variables correspond to the voltage membrane of the neuron v and a recovery variable u. There are four dimensionless parameters (a, b, c, d) and an input current $I(t)$. The spike mechanism works by resetting variables v and u when the voltage reaches some fixed value. Figure 2 shows the neuron spikes in the v variable and the phase portrait of the two dynamical variables. The second plot also presents the nullclines of the system, *i. e.* the curves where only one of the differential equations is equal to zero.

The current $I(t)$ includes all the synaptic currents. The neuron may receive synaptic inputs from other neurons. When one of these neuron fires a spike, after some time delay T, a post-synaptic current pulse is added to $I(t)$: for an excitatory (inhibitory) synapse a step increase (decrease) of size W is followed by an exponential decay towards the previous current value with a time constant

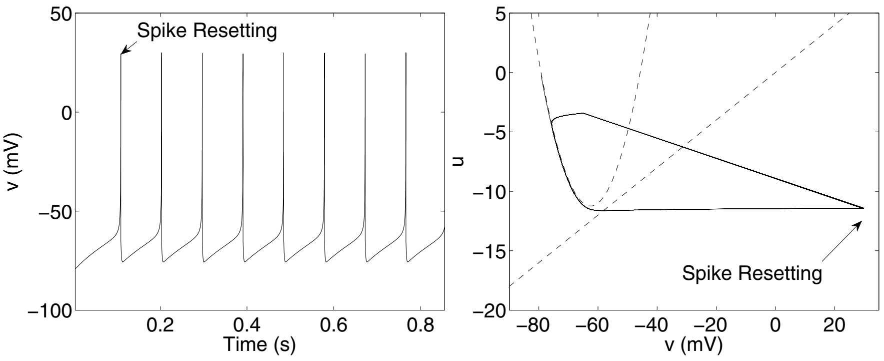


Fig. 2. Neuron membrane voltage trace v (left) and phase portrait u, v (right). The dashed lines represent the nullclines of the dynamical system and the full line is the periodic spiking orbit showing a limit cycle.

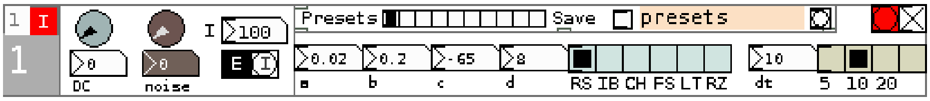


Fig. 3. Neuron Module. The numbers in the left indicate MachineID and Instrument number respectively. There are two sub-modules: local input current and neuron parameters with a preset manager.

t_s (which is also an adjustable parameter). In addition to the synaptic currents, $I(t)$ includes other currents that can be controlled by the user: a DC current, a zero-average noisy current, and excitatory or inhibitory pulses.

This model has the advantage of being both computationally cheap and versatile. With simple adjustment of its parameters, it can reproduces several known neuronal spiking patterns such as bursting, chattering, or regular spiking. For a complete description of the possible behaviors see [10].

The user can generate any number of neuron modules (figure 3), only limited by CPU capability. Each module has local input currents, and can also be affected by a global current and noise level sent from the main panel. The user can choose between six presets of firing modes or manually set the model parameters. Every neuron module has a particular integration time, allowing different *tempos* for equal firing modes and parameter settings. There is also a presets manager for saving and loading stored combinations of parameters.

Network. This structure allows designing and configuring the network by selecting synaptic coupling between neurons, and the delays. In the left column the user can select the neuron type, switching between excitatory (E) and inhibitory (I) post-synaptic pulses. The central module of this structure is the *connectivity matrix* (figure 4). Delays are set in the dark gray number boxes and synaptic weight have color depending on the E/I status, so at a glance is easy to grasp the network status. A synaptic weight equal to zero means that there is no synaptic connection. There is number box (flat) for setting the same synaptic weights for all the connections, and also a random set up generator.

Instruments. In this structure the user can map the neural activity to sound event generation and sound event parameters. Each instrument has basic parameters the user may expect to control. In the example shown in figure 5, Instrument 1 of Machine (ID) 1 has four sub-modules mapping: rhythm, duration, pitch and intensity. Every one of this has a Machine ID number and a Neuron number from which spikes and firing rates will be mapped. Both variables are by default initialized with the same Machine ID and Instrument number. (A spatialization sub-module (not shown) allows the instrument to control the distribution of events in a multichannel system.) The initial setup this instrument module works with a basic synth oscillator or sends MIDI messages.

When the parameters are switched to manual (M) the user can fix the sound event parameters to any desired value. When switched to automatic (A), it maps

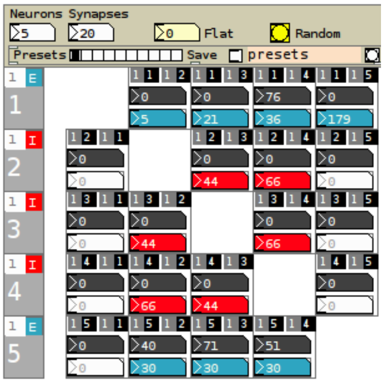


Fig. 4. Network connectivity matrix. This structure allows designing the network by selecting synaptic gains (blue and red) and delays (dark gray) between neurons. In the left column the user switches between excitatory and inhibitory postsynaptic pulse for every neuron.

the firing rate of the input neuron to a modulation of the corresponding sound event parameter. In the pitch case, the firing rate value is used for frequency modulation, with a movable offset and configurable depth. Intensity varies between 0 and 1 for each instrument, and the overall volume is set from the main panel.

Santiago has currently two visualization panels where network activity is depicted in real-time: spike view and event view. In spike view all the individual traces of v are plotted in rows. The event view shows vertical lines in rows (one for each neuron), corresponding to spikes, discarding sub-threshold activity.

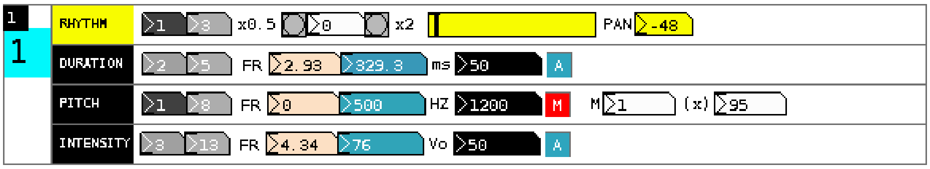


Fig. 5. Instrument module. There are four sub-modules that map rhythm, duration, pitch and intensity from neurons.

3 Network Activity

There are several ways to set up a network, depending on the desired goal. One possible way is to build simple feed-forward circuits that perform basic logic operations, such as coincidence detection (integrator neuron) or filtering intervals (resonator neuron). Other ways are to use mutual inhibition or excitation, or closed loops in order to achieve collective oscillations, like in CPG networks or

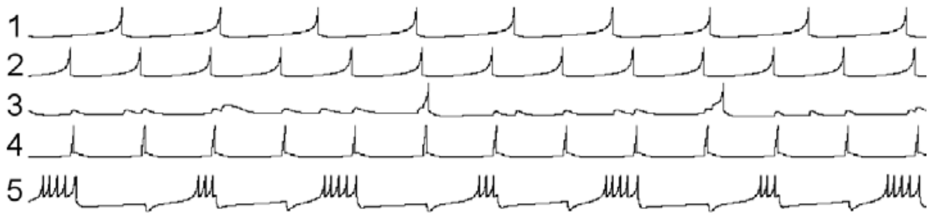


Fig. 6. Spike view of the network displayed in figure 1. Neurons 1 and 2 are Regular Spiking (RS) neurons, and both excite neuron 3, also RS. When their spikes are close enough in time, they make neuron 3 spike, acting as a coincidence detector. Neuron 4 is a Low Threshold neuron excited by neuron 2 and spikes every time neuron 2 does. At the same time, it inhibits 5 which has a chattering behavior.

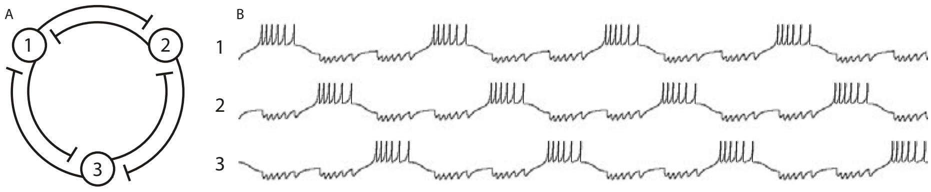


Fig. 7. Network diagram (A), and spike view (B) of a fully connected inhibition network working as a CPG. Notice the rhythm and periodicity of the pattern.

coupled oscillators. Here we present three different examples: a simple network with a coincidence detector, an elementary CPG and a small network more oriented to obtain musical output.

As our basic example we used the same network displayed in figure 1. In this circuit, neuron 3 detects concurrent spikes from 1 and 2. Neuron 4 acts as an inhibitory inter-neuron for neuron 2 and inhibits the repetitive bursting pattern from neuron 5. The outcome of this network is the repetitive pattern displayed in figure 6).The length of the cycle varies depending on the parameters used.

We also tested whether our environment could reproduce some elementary CPGs, such as the inhibition network studied in [12]. *SANTIAGO* was able to reproduce the characteristic sustained oscillations patterns of the model. In figure 7 we show the burst of three chattering neurons with mutual inhibition. This example is particularly interesting to use with a spatialization module, because we can have travelling patterns in space.

In our last example, we designed a network intended to depict a complex behavior in the sense mentioned in the introduction. Further, we were interested in obtaining, as an output of a single instrument associated to that network, a simple musical phrase with melodic and dynamic plasticity. For this purpose, we devised a small, but densely connected network of three chattering inhibitory neurons and two excitatory neurons (one chattering and one regular spiking)

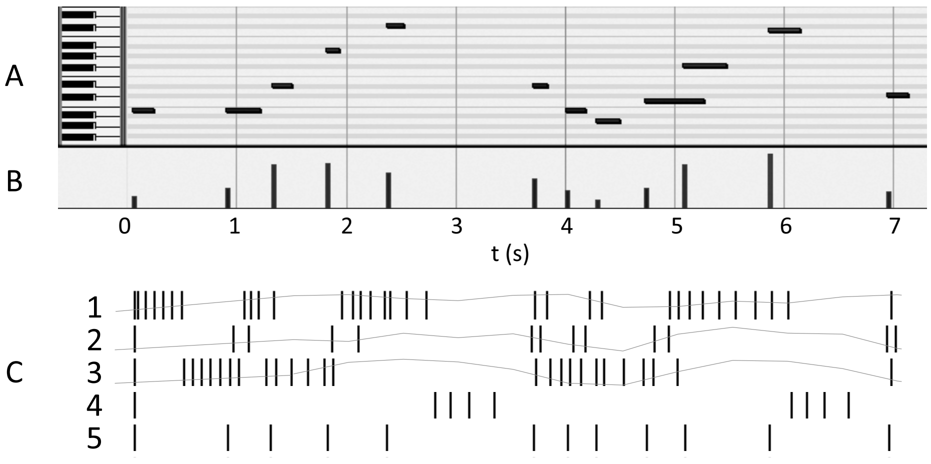


Fig. 8. An example of a musical score generated by a five-neuron network. Piano roll (A) and note velocity (B) show data received through a virtual MIDI port into an external sequencer. Event view (C) of the five-neuron network. Events from neuron 5 were selected to trigger the notes. The duration, intensity and pitch were modulated with the firing rate of neurons 1, 2 and 3 respectively. The delayed firing rates are plotted as thin lines in the corresponding row.

which produced a nearly recurring temporal pattern that shows up with slight variations on each cycle, as displayed in figure 8. The notes were triggered by neuron 5. The length of the notes was controlled by the firing rate of neuron 1, the intensity with the firing rate of neuron 2 and the pitch with neuron 4. The firing rate is computed with a time window of two seconds, inserting a delay between the events onset time and the corresponding firing rate computation.

4 Conclusions

We presented a real-time environment for generative music creation, based on the intrinsic dynamics of biological neural networks. From this perspective, several mapping strategies were presented towards a composition-performance technique. *SANTIAGO* is a system that allows the user to compose and perform music in which every sound event and parameter may interact with the others. The presence of an event may alter the whole network and a change in a bigger scale may be reflected in a micro level too. This brings together the essence of complex systems and music performance. We can think of small musical elements with simple rules which scale to another level when they interact. For more references, project updates, and audible examples go to http://lapso.org/santiago

References

1. Danks, M.: Real-time image and video processing in Gem. In: Proceedings of the International Computer Music Conference, pp. 220–223. International Computer Music Association, San Francisco (1997)
2. Destexhe, A., Marder, E.: Plasticity in single neuron and circuit computations. Nature 431, 789–795 (2004)
3. Eldridge, A.C.: Neural Oscillator Synthesis: generating adaptative signals with a continuous-time nerual model
4. Franklin, J.A.: Recurrent neural networks for music computation. INFORMS Journal on Computing 18(3), 312 (2006)
5. Galanter, P.: What is generative art? Complexity theory as a context for art theory. In: GA2003–6th Generative Art Conference (2003)
6. Gribbin, J.: Deep Simplicity: Bringing Order to Chaos and Complexity. Random House (2005)
7. Hooper, S.L.: Central Pattern Generators. Embryonic ELS (1999)
8. Huron, D.: Sweet anticipation: Music and the psychology of expectation. MIT Press, Cambridge (2008)
9. Izhikevich, E.M.: Simple model of spiking neurons. IEEE Transactions on Neural Networks 14(6), 1569–1572 (2004)
10. Izhikevich, E.M.: Dynamical systems in neuroscience: The geometry of excitability and bursting. The MIT press, Cambridge (2007)
11. Kocho, K., Segev, I.: The role of single neurons in information processing. Nature Neuroscience 3, 1171–1177 (2000)
12. Matsuoka, K.: Sustained oscillations generated by mutually inhibiting neurons with adaptation. Biological Cybernetics 52(6), 367–376 (1985)
13. Molnár, G., et al.: Complex Events Initiated by Individual Spikes in the Human Cerebral Cortex. PLoS Biol. 6(9), e222 (2008)
14. Mozer, M.C.: Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. Connection Science 6(2), 247–280 (1994)
15. Puckette, M.S.: Pure Data. In: Proceedings, International Computer Music Conference, pp. 269–272. International Computer Music Association, San Francisco (1996)
16. Soltau, H., Schultz, T., Westphal, M., Waibel, A.: Recognition of music types. In: Proceedings of the 1998 IEEE International, vol. 2, pp. 1137–1140. IEEE, Los Alamitos (2002)
17. Taylor, I., Greenhough, M.: Modelling pitch perception with adaptive resonance theory artificial neural networks. Connection Science 6(6), 135–154 (1994)
18. Yadegari, S.: Chaotic Signal Synthesis with Real-Time Control - Solving Differential Equations in Pd, Max-MSP, and JMax. In: Proceedings of the 6th International Conference on Digital Audio Effects (DAFx 2003), London (2003)

Neurogranular Synthesis: Granular Synthesis Controlled by a Pulse-Coupled Network of Spiking Neurons

Kevin McCracken, John Matthias, and Eduardo Miranda

Faculty of Arts, University of Plymouth, UK
{kevin.mccracken,john.matthias,eduardo.miranda}@plymouth.ac.uk

Abstract. We introduce a method of generating grain parameters of a granular synthesiser in real-time by using a network of artificial spiking neurons, the behaviour of which is determined by user-control of a small number of network parameters; 'Neurogranular synthesis'. The artificial network can exhibit a wide variety of behaviour from loosely correlated to highly synchronised, which can produce interesting sonic results, particularly with regard to rhythmic textures.

Keywords: Spiking neurons, granular synthesis, interactive musical control systems.

1 Introduction

A recent development in the field of autonomous interactive musical control systems [13], which has received a great deal of attention [3] [6] [15], utilises the adaptive nature of an artificial recurrent network of nonlinear spiking neurons [14]. Several methodologies have been developed in which sonic events are triggered by neuronal firing in a network of nonlinear Integrate-and-Fire (IF) neurons which have been applied in various artistic contexts [5][12]. These firing events are attractive from a musical point of view for several reasons; the collective temporal dynamics of firing occur within a natural temporal scale for musical phrases, they have the potential to be controlled by external stimulation by a musician and can adapt and change according to relationships between external stimulation, internal 'noisy' currents and plasticity within the network of neurons.

In this paper, we focus on the neuronal control of a granular synthesis engine, in which grains of synthesised sound are triggered by neuronal firing events in a simple spiking network model [8]. We will introduce granular synthesis, develop a rationale for the control of a granular synthesis engine using artificial neural networks and introduce a prototypical model, which we intend to develop in further work.

2 Granular Synthesis

Granular synthesis is a method for generating sound using a series of audio 'grains', typically of a few tens of milliseconds in duration [17]. Each grain is usually an

© Springer-Verlag Berlin Heidelberg 2011

envelope-modulated waveform (as shown in fig. 1) and the grain duration, envelope function, amplitude, pan, waveform, frequency, etc. must be specified as control data. The grains may be temporally isolated or may overlap. The grain envelope removes clicks and glitches caused by discontinuities in the audio waveforms produced at the grain boundaries. The form of this grain envelope has a considerable effect upon the character of the resultant sound comprised of sequences or clouds of multiple grains.

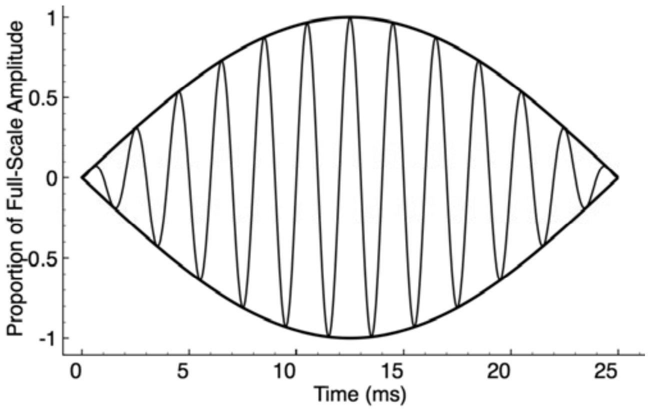


Fig. 1. A 'grain' is an envelope-modulated tone or 'chirp', typically a few tens of milliseconds in duration

All of the parameters mentioned above constitute control data, which influence the perceived sound. Since each grain can have a dozen or more parameters associated with it and since grain durations can be as little as 10ms, the necessary control data rate to realise granular synthesis is quite large. It is not possible for a human operator to produce this amount of control data in real-time, so it must be generated, either by some sort of deterministic algorithm (set of instructions/equations) or a non-deterministic stochastic process.

Most current granular synthesis employs stochastic processes to generate grain control data [17]. However, granular synthesis with stochastically generated grain parameters will necessarily tend to produce 'noisy' synthesis and greatly limit the controllability of any system generating control information in this manner. A grain control generation system producing output data, which is, to some degree, correlated with previous data, is likely to produce more interesting and flexible results, particularly in terms of temporal information. To move forward with granular synthesis, novel methods of generating real-time grain control data are likely to prove fruitful.

3 Research Background

One of the main driving forces behind this work is that the level of synchronisation in nonlinear distributed systems is controlled by the strength of individual interactions. A desirable temporal output for granular synthesis in music would be a controllable signal, which lies at the border between randomness and synchrony. Two-dimensional reduced spiking neuronal models (reduced from the four-dimensional Hodgkin-Huxley model [7]) have been shown recently to self-organise at this boundary [11]. Recent artistic projects, The Fragmented Orchestra [5] and the Neurogranular Sampler [5][14] focused on the two-dimensional Izhikevich spiking network model [8] and trigger sound samples upon neuronal firing. In this work, we introduce a Neurogranular Synthesiser, which triggers the synthesised grains upon neuronal firing within the simple Izhikevich network.

3.1 Why Izhikevich Spiking Neurons?

Any one of a number of types of artificial neural network are possible candidates for real-time control of a granular synthesiser, such as McCulloch-Pitts, Hopfield or back-propagation networks, self-organising maps, Multi-Layer Perceptrons (MLP), Radial Basic Function (RBF) networks and, indeed, other spiking neuron models, such as IF neurons. The Izhikevich neuron was chosen for the following reasons;

1) behavioural richness; real neurons exhibit a diverse range of firing behaviour and the Izhikevich model's biological plausibility provides a similarly rich behavioural palette to be mapped to grain parameters.
2) dynamical richness; Spiking Neural Networks (SNNs) in particular manifest a rich dynamical behaviour. Transmission delays between neurons can give rise to separate functional groups of neurons (polychronisation) with the potential to be harnessed when mapping to grain parameters.
3) spiking; certain useful behavioural features require spiking behaviour, e.g. Spike Timing-Dependent Plasticity (STDP), discussed below.
4) computational efficiency; the Izhikevich model was considered an excellent trade-off between accuracy of biological triggering behaviour simulation and the amount of computer processing to implement it [9].
5) We know from our own 'experience' that networks of real spiking neurons can produce a rich and adaptive behaviour.

3.2 The Izhikevich Neuron Model

The Izhikevich neuron model is defined by the following pair of first-order differential equations [8];

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + i \, . \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \, . \tag{2}$$

and the reset condition when v ≥ +30mV;

$$\begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} . \tag{3}$$

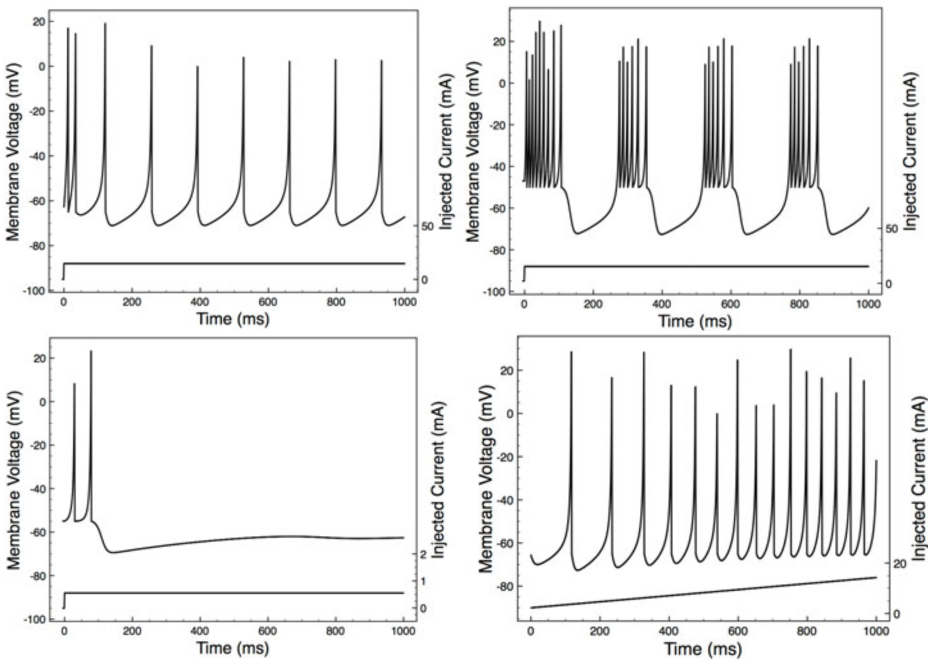


Fig. 2. Some Izhikevich behaviours produced by a single neuronet neuron. Top left to right bottom; tonic spiking (TS), tonic bursting, phasic bursting & TS firing freq. vs. input stimulus.

The dependent variables are the recovery variable u and the cell membrane voltage v, respectively. The behaviour of an individual Izhikevich neuron is determined by the four neuron parameters (a, b, c and d) and the input stimulus, i. The mammalian cortex contains many different types of neuron whose spiking behaviour differs in response to an input stimulus. Appropriate selection of these four parameters models the full range of these behaviours, some of which are illustrated in fig. 2.

3.3 Networks of Izhikevich Neurons

The fact that the four Izhikevich neuron parameters in conjunction with the input stimulus can produce such a wide variety of spiking behaviour [8] and that spike and grain timescales are of the same order, makes the Izhikevich model appropriate for real-time control of a granular synthesiser.

When artificial neurons are interconnected in a network, with each receiving inputs from many others, their collective behaviour becomes more complex and interesting. Connections to other neurons (or synapses) vary in strength and type. There are two types of synapses: excitatory, which increase membrane potential in response to a positive synaptic message and inhibitory, which decrease it. The ratio of excitatory to inhibitory neurons in a mammalian cortex is approximately 4:1. Synapses are simulated in the network of N neurons discussed below via an N×N synaptic connection weight (SCW) matrix S with elements $s_{i,j}$, such that firing of the jth neuron instantaneously changes the membrane voltage of the ith neuron v_i by $s_{i,j}$, where;

$$s_{i,j} = S_e c_{i,j} : 1 \le i, j \le N, S_e = 0.5 . \tag{4}$$

for excitatory neurons and

$$s_{i,j} = S_i c_{i,j} : 1 \le i, j \le N, S_i = -1.0 . \tag{5}$$

for inhibitory neurons. The parameters S_e and S_i are weighting factors for excitatory and inhibitory synapses respectively and $c_{i,j}$ are random variables such that;

$$c_{i,j} \sim U(0,1) \tag{6}$$

For each neuron in the network, the SCW weights from all connected neurons which have fired at a given instant are summed and added to the input stimulus, i.e. at each instant of time t, the total input I_i (t) to the ith neuron in the network,

$$I_i(t) = i_i(t) + \sum_{j=fired}^{N} s_{i,j} . \tag{7}$$

where i_i (t) is the input stimulus to the ith neuron at time t.

3.4 Propagation Delays, STDP and Spike Coding Schema

Electrochemical signals in biological neural networks are transmitted between neurons at speeds of the order of a few metres per second, giving rise to time differences between sent presynaptic and received postsynaptic signals. These propagation delays range from fractions of a millisecond to a few tens of milliseconds and are responsible for polychronisation [10]. The research presented here employs propagation delay as a prospective tool in mapping behavioural and dynamical features to grain parameters in a granular synthesiser.

Research from a number of sources has shown that synaptic connection strengths are modulated by relative timing between pre-synaptic action potentials and post-synaptic firing. Within a time difference of around 10-50 ms, long-term synaptic connection strengths are increased when pre-synaptic action potentials precede post-synaptic firing and decreased when they follow post-synaptic firing [1]. Such a mechanism is termed Spike Timing-dependent Plasticity (STDP) and is a means of learning a response to a particular stimulus. STDP will be implemented in the final version of the neurogranular synthesiser described below.

The method by which spike trains are translated into meaningful grain control parameters can greatly influence the resultant sound of a granular synthesiser driven by them. Possible spike-train coding schemas are as follows;

1) temporal coding - a spike could simply trigger the production of a grain
2) rate coding – the spike count per unit time could be used to generate parameter values; i) for a single neuron, or ii) for a group.
3) spike coding – relative spike times of a group of neurons could be scaled to generate a grain parameter value, i) as time to first spike, or ii) phase relative to stimulus for a group.

The efficacy of possible spike-coding schema in mapping spiking behaviour to grain parameters (in terms of timbral malleability) will be evaluated in future work.

4 Research Objectives

The first neurogranular synthesiser was *Neurosynth* [16]; a real-time implementation of neurogranular synthesis using 5 networks of 10 identical Izhikevich neurons, each triggering a half sine-wave windowed sine oscillator, the frequency of which is determined via rate-coding. The 5 network output grain streams are summed to produce the final audio output. The four Izhikevich parameters for each network are available for user control, as are the grain amplitude, frequency, duration and density.

The research will address the limitations of *Neurosynth* and extend the scope of neural network control and modularity of the neurogranular approach, allowing the user to configure parameters such as the following;

- number of neurons in the network
- mappings between oscillators and neurons
- inhibitory neuron proportion/arrangement
- network constitution from homogeneous (with identical neurons) to heterogeneous (with neurons of varied behaviour)
- topology or interconnectivity of the neurons
- propagation delays
- Spike Timing Dependent Plasticity (STDP)
- multiple oscillator waveforms
- multiple grain envelope functions

5 Building a Neurogranular Synthesiser

The neurogranular synthesiser was coded in the C language as two interconnected real-time external objects for Cycling74's™ MAX/MSP audio development environment using Cycling74's™ Applications Programming Interface (API): the neural network neuronet and audio grain generator multigrain~. The neuronet SNN objects output passes scheduling and grain parameter data to multigrain~'s input.

5.1 The Grain Generator Object

The multigrain~ object accepts an argument specifying the required polyphony, P. Each GG produces a 'chirp' of specified waveform, frequency, amplitude, duration, pan and grain envelope shape when a properly formatted message is received at the input. At the time of writing, a basic one-to-one prototypical mapping of neurons to grains has been implemented. Currently, the grain oscillators have fixed frequencies derived from P equal subdivisions of a five-octave range and all other grain parameters are fixed. In future versions of the objects, much greater flexibility in the mapping from neurons to grains and codification of frequency and scheduling will be implemented. Each GG produces non-overlapping grains constituting 'a grain stream'. All streams are mixed to produce the output audio. The following grain parameters are provided for each grain by the triggering input message;

- waveform (sine, triangle, sawtooth, square or white noise)
- frequency
- grain amplitude (0.0 to 1.0)

- grain envelope (rectangular, triangular, trapezoidal, or Tukey)
- grain envelope fade-in slope (trapezoidal and Tukey waveforms only)
- grain duration (10-100ms)
- pan left to right (-50 to +50)

5.2 The Izhikevich Network Object

The neuronet object simulates a real-time N-neuron network of Izhikevich artificial neurons, N being accepted as an argument. Neuronet outputs data to control the multinet~ grain generator and has four inputs for real-time control of the Izhikevich parameters a, b, c and d from which individual neuron parameters are derived. On instantiating a neuronet object of N neurons, the Izhikevich parameters are initially randomised to produce a spread of neuron behaviours. The network can be made homogeneous, with all neuron parameters being identical, or heterogeneous, with individual neuronal parameters for the Izhikevich model being statistically distributed around the four user-supplied average network parameter values (a, b, c and d). Any user adjustment of the Izhikevich parameter magnitudes (a, b, c and d) input to the neuronet object, moves the statistical centre of those parameters through the phase-space for every neuron in the network, so modifying the spiking behaviour of the entire network.

The leftmost input also accepts other real-time messages to configure network parameters such as propagation delay modelling, network topology, constitution (homogeneity or heterogeneity) and input stimulus, etc. On instantiation, a matrix of propagation delay times is calculated from each neuron in the network to every other neuron. At the time of writing, the synaptic connection strengths stored in the SCW matrix remain static after initial randomisation, but future versions will implement STDP by allowing the SCW matrix elements to be varied in real-time.

6 Discussion

The first implementations of granular synthesis were ‘offline’ systems which did not operate in real time. Pioneering work by Iannis Xenaxis combined music concrète approaches and Dennis Gabor’s ideas on audio quanta [4] to realise audio grains with many short sections of pre-recorded sound on magnetic tape. Curtis Roads and John Alexander’s Cloud Generator program [18] and Barry Truax’s GSX and GSAMX programs [19] were the first computer-based implementations of granular synthesis. Cloud Generator is an offline (non-real-time) system providing granular synthesis and sampling. All parameters are determined with either linear functions between user-specified start and end points, statistical algorithms with user-specified maxima and minima, or random processes and the audio output is rendered as a sound file. Barry Truax’s GSX and GSAMX programs [19] perform granular synthesis in real-time with a polyphony of 20 voices. Grain control parameters changed via single keyboard strokes are either fixed values, linear functions (or ramps), or statistically derived around a fixed mean value. More recent examples of real-time granular synthesis include Stochos [2] and Propellerheads'™ Maelström graintable synthesiser. Both Stochos and Maelström employ wavetable-based oscillators and numerous grain control parameters are generated in real-time by deterministic or statistical functions. In all cases, setting bounds for these functions for numerous grain parameters constitutes a control interface of some complexity.

For any synthesis methodology, one of the most important issues, when designing an instrument, is managing the complexity of the controlling parameters to provide an interface which responds in an intuitive manner to the composer. Granular synthesis, in particular, requires the manipulation of a very large number of grain parameters. The majority of approaches utilise combinations of deterministic and statistical functions to generate the necessary grain control parameters in real-time, but still require the user to specify many parameters.

Using neural networks to generate grain parameters has the potential to greatly simplify the user interface of a granular system. The Izhikevich model of artificial neuron employed has a behaviour which is determined by four parameters and the behaviour of every neuron in the neuronet object's is derived from these four parameters (in a homogeneous network, the Izhikevich parameters of each neuron are identical and, in a heterogeneous network, the parameters of each neuron are derived as a statistical variation around those four values). Thus, the behaviour of the whole network can be radically altered by changing just four parameters. In fact, this could be further simplified once musically useful behaviours are mapped to regions in the network parameter space so that, potentially, one controller might be varied in real-time to allow the user to move through all of the musically 'useful' network behaviours. The input stimulus to the network influences network firing rate and the nature of that stimulus influences how correlated network firing is. Thus, for example, a simple 2-dimensional controller could provide effective adjustment of grain density and character with the y axis controlling amplitude and the x axis varying the nature of the input continuously from constant to random.

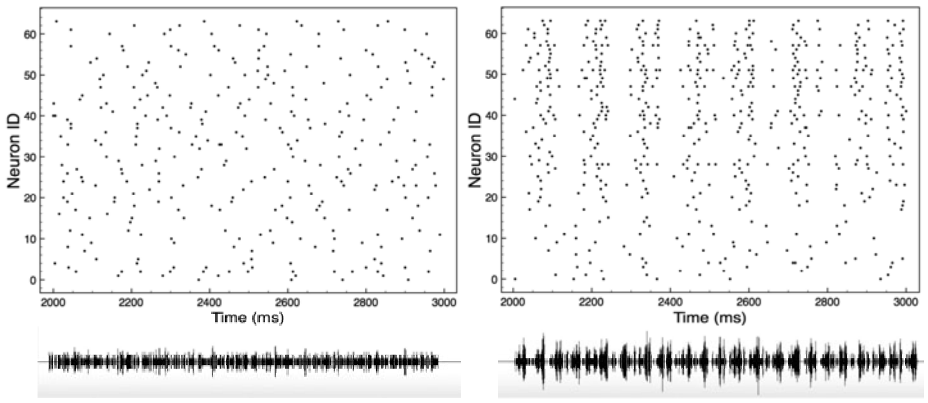


Fig. 3. Raster plots and output waveforms from neuronet driving multigrain~ for a heterogeneous 64-neuron network with input noise without (left) and with (right) delays

7 Conclusion

The prototype neuronet AN object has demonstrated potential for real-time granular synthesiser control, but also the need for further work to optimise its sonic potential. The current version is limited in scope (although interesting rhythmically) as network firing simply schedules grains; this basic implementation of spike-coding will be addressed in future work. Although the synthesiser engine (multigrain~) is capable of generating five different waveforms, four different grain envelopes of grain durations from 10-100ms and harnessing panning and amplitude data for each grain, this has not been exploited at the time of writing of this paper. Neural network-derived spike coding of some or all of these parameters could give very interesting sonic results. For instance, a harmonically rich waveform could be low-pass filtered with cut-off frequency modulated by the firing rate of a functional neuron group, or the same group firing rate might determine proportions of the fundamental and a number of harmonics, a higher firing rate equating to a higher high-frequency harmonic content, etc. Future versions of the described objects will incorporate plasticity to allow the sonic potential of topographical considerations and stimulus-driven network adaptation to be investigated. The intent of this research is to produce a musically intuitive instrument, which provides considerable real-time player control of timbre. Much work needs to be done on the most tonally appropriate means of mapping network spiking behaviour to grain parameter assignment. Further refinements to the system will maximise grain polyphony via code optimisation and dynamic oscillator resource allocation and a simple, intuitive user-interface will need to be developed. A device such as an iPad, capable of tracking multiple (half-a-dozen, or more) finder-tip positions, might serve as an excellent finger-actuated grain parameter controller.

References

1. Bi, G., Poo, M.: Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type. Journal of Neuroscience 18(24), 10464–10472 (1998)
2. Bokesoy, S., Pape, G.: Stochos: Software for Real-Time Synthesis of Stochastic Music. Computer Music Journal 27(3), 33–43 (2003)
3. Ferguson, E.: The Future of Music? It might be... The Observer, London (February 25, 2009)
4. Gabor, D.: Acoustical Quanta and the Theory of Hearing. Nature 159(4044), 591–594 (1947)
5. Grant, J., Matthias, J.R., Ryan, N., Jones, D., Hodgson, T., Outram, N.: The Fragmented Orchestra (2009), `http://www.thefragmentedorchestra.com`
6. Hickling, A.: Noises Off, The Guardian (December 19, 2008)
7. Hodgkin, A.L., Huxley, A.F.: A Quantitative Description of Ion Currents and its Applications to Conduction and Excitation in Nerve Membranes. J. Physiol (Lond.) 117, 500–544 (1952)
8. Izhikevich, E.M.: A Simple Model of Spiking Neurons. IEEE Transactions on Neural Networks 14, 1469 (2003)
9. Izhikevich, E.M.: Which Model to Use for Cortical Spiking Neurons? IEEE Transactions on Neural Networks 15, 1063–1070 (2004)

10. Izhikevich, E.M., Gally, J.A., Edelman, G.M.: Spike-timing Dynamics of Neuronal Groups. Cerebral Cortex 14, 933–944 (2004)
11. Lubenov, E.V., Siapas, A.G.: Neuron. 58, 118–131 (2008)
12. Matthias, J.R., Ryan, N.: Cortical Songs. CD. Nonclassical Records, London (2008)
13. Miranda, E.R., Wanderley, M.M.: New Digital Musical Instruments: Control and Interaction Beyond the Keyboard. The Computer Music and Digital Audio Series. A-R Editions, Middleton (2006)
14. Miranda, E.R., Matthias, J.R.: Music Neurotechnology for Sound Synthesis. Leonardo 42, 439–442 (2009)
15. Morrison, R.: Liverpool ends year on a cultural high with The Fragmented Orchestra. The Times, London (2008)
16. Murray, J., Miranda, E.R., Matthias, J.: Real-time Granular Synthesis with Spiking Neurons. In: Proceedings of Consciousness Reframed - 8th International Conference. University of Plymouth, Plymouth (2006) (invited talk)
17. Roads, C.: Microsound, pp. 14–16. MIT Press, Cambridge (2004)
18. Roads, C.: Microsound, pp. 383–388. MIT Press, Cambridge (2004)
19. Truax, B.: Real-Time Granular Synthesis with the DMX-100. Computer Music Journal 12(2) (1988)

Interactive Biomimetic Space: An Interactive Installation to Explore Living Architecture

Liraz Mor[1], Chao Liu[1], and Sebastian von Mammen[2]

[1] Computational Media Design
[2] Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
{lmor,liuchao,s.vonmammen}@ucalgary.ca

Abstract. This paper describes a computer-based Interactive Biomimetic Space (IBS) installation. Through an interactive process, a user creates and relates new spaces. The simulation of a simplified ecosystem populates the created space with life. The installation attempts to inspire architectural design ideas by integrating biological principles. In particular, the biological concepts of stochastic motion, interaction and reproduction of artificial swarms have been explored and applied. Both the user and the swarm agents contribute to the design process, which results in interesting design outputs with a certain degree of unpredictability. We introduce the theoretical background of our work, outline its technical implementation, and present various interaction examples and scenarios.

Keywords: Interactive art, architectural design, swarm dynamics, computational development.

1 Introduction

With the development of various Computer Aided Design (CAD) techniques, many designers and artists today understand the computer as a tool for facilitating design processes and art creation. Computer technology is utilized not only as an effective approach to present final works but also as a practical assistance in formulating design ideas and inspiring innovative concepts. Computer scientists and biologists have developed numerous computational models of systems that retrace phenomena observed in nature [1,5]. Such bio-inspired models have the potential to provide solutions for challenges in other disciplines. In particular, they provide designers and artists with a new perspective to develop their aesthetic creation [2,4,6,7].

In this work, we present an Interactive Biomimetic Space (IBS) installation that utilizes several bio-inspired computational concepts to explore living architecture. Specifically, within architectural space that is initialized by the user, a swarm of agents are introduced as the actual inhabitants. Their habitat then grows gradually through the coordination and collaboration among the agents themselves and the interaction processes between the agents and the user.

© Springer-Verlag Berlin Heidelberg 2011

Section 2 gives a brief review on the related works. Section 3 details the implementation and setup of the IBS installation and illustrates the interaction scenarios. Section 4 focuses on the design results. We conclude with an outlook on possible future works.

2 Related Work

The flocking behaviors of social animals, such as in flocks of birds or schools of fish, can be simulated by means of boids [5]. Given a set of simple interaction rules, such as *separation*, *alignment*, and *cohesion*, the boids simulation can be rather complex due to the large number of interaction events among the participating agents. By adding additional interaction rules to the original Boids implementation, the user can be involved in these interaction processes. Unemi and Bisig, for instance, developed the Flocking Orchestra by adding two forces to support human-swarm interactions: “an attractive force causes the agents to move towards the front part of the virtual world when they perceive visitor motion; a repellant force pushes the agents away from the front part in the absence of any visitor’s motion” [6]. In the Flocking Orchestra, each agent controls a MIDI instrument and plays a music note according to its state. As the user varies his gestures or positions in front of the camera, the agents’ states will be adjusted correspondingly, resulting in the generation of music. This example demonstrates how Boids can be harnessed for an artistic creation via human-swarm interactions.

Swarm Intelligence describes the collective behaviors of decentralized and self-organized systems, such as ant colonies [1]. There is no central leader or hierarchical command structure within the system to govern the individual agents’ behaviors. Instead, the agents interact locally with one another and with their surroundings based on simple rules. Jacob et al. explored Swarm Intelligence in their SwarmArt project [2]. By means of video cameras and computer vision algorithms, the interaction between a user and the swarm agents was realized. As a result, a dynamic and evolving swarm system was developed as a new tool for artists. von Mammen and Jacob applied Swarm Intelligence to inspire architectural design by introducing the Swarm Grammar system [7]. This system allows agents to reproduce, communicate and build construction elements during simulations. Different architectural shapes emerge from the interaction processes of swarms with different behaviors. By breeding swarms through evolutionary algorithms, the system produces unexpected, creative architectural models.

Niche construction is the process in which an organism alters the environment to increase its own and its offspring’s chance for survival [3]. McCormack and Bown applied principles of niche construction to an agent-based line drawing program to increase its output diversity [4]. An allele in the agent’s genome is utilized to describe the agent’s preference for the density of drawn lines in its environment. A favorable environment increases the probability of agent reproduction. In this way, parents may construct a niche and pass on a heritable environment well-suited to their offspring.

3 The IBS Installation

Inspired by the works described above, the IBS installation is a simple, computer-based interactive ecosystem, inhabited by swarm agents. Its motivation is the exploration of architectural design processes in and for complex, lively environments. A user engages with these swarm agents in a process of unfolding architectural spaces. Dynamically adding simple rooms to accommodate the demands of the swarm and the desires of the user, this collaborative effort yields interesting artificial design scenarios.

The user sees three windows on a computer screen (Fig. 1). The main window shows the IBS scene comprising the user's agent and the swarm agents, and displays the dynamic IBS in real-time. In the upper-right corner of the screen, a motion detection window is displayed to help the user to be aware of his motions in front of a camera. A 2D map window is located in the upper-left corner of the screen to assist the user in navigating through the scene. By default, the main window shows the scene in 3D mode.

In the main window, the user, or audience, sees different graphical elements. The blue sphere represents a particular agent which is controlled by a motion detection system. Other smaller spheres might be shown that represent autonomous swarm agents that dwell in the simulated space. The IBS is made of individual rooms that are depicted as transparent cubes. The relationships among the three interactive components—user, agents, and IBS—are displayed in Fig. 2. Generally, the user creates the first several rooms and introduces a flock of swarm agents. Subsequently, according to the characteristics of the rooms, such as the room size and the room temperature, swarm agents are triggered to interact with their partners as well as the environment.

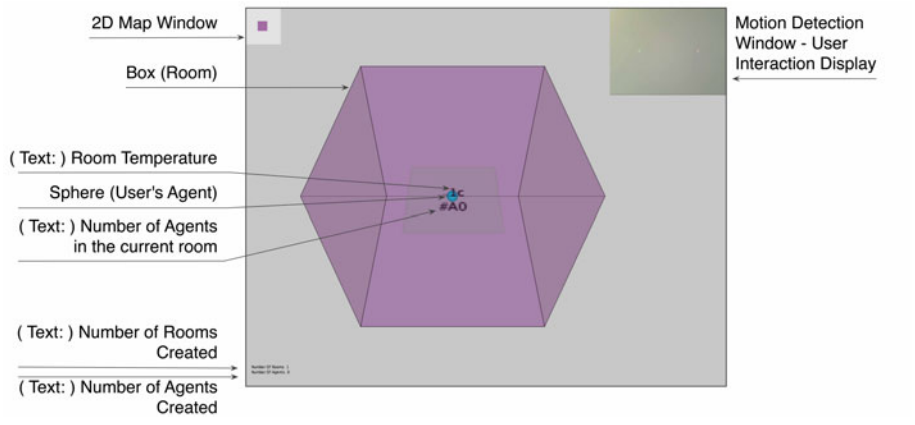

2D Map Window
Box (Room)
(Text:) Room Temperature
Sphere (User's Agent)
(Text:) Number of Agents in the current room
(Text:) Number of Rooms Created
(Text:) Number of Agents Created
Motion Detection Window - User Interaction Display
1c
#A0

Fig. 1. In addition to the main view, the IBS installation displays a map view in the top-left and a user interaction display in the upper-right corner

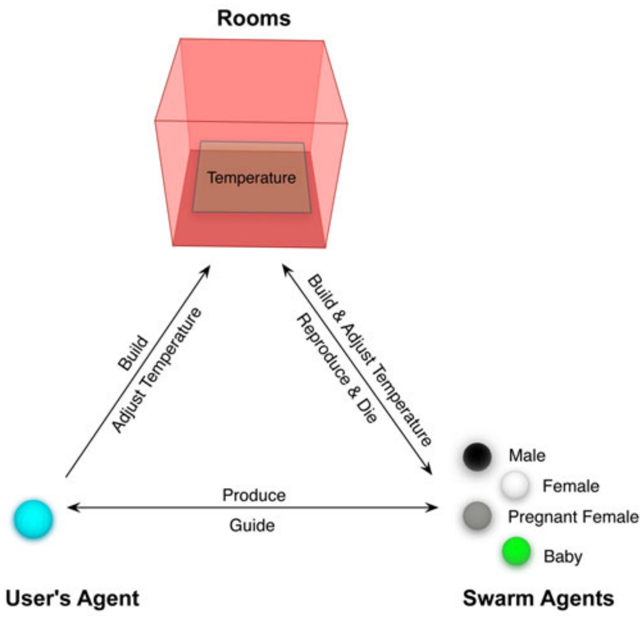


Fig. 2. The relationship triangle among three components: User's Agent, Swarm Agents and Rooms

3.1 The User's Agent

A basic requirement for a meaningful simulation is a set of tools to direct and constrain the swarm agents to explore the architectural design space. We choose to implement that set of tools as a user's agent. The user's agent represents the user in the system and is controlled by him. It is rendered as a blue sphere, and its abilities are different then those of other swarm agents. The user's agent functions as an architect, coordinator and observer. It is responsible for shaping the initial space and leading the swarm agents to develop their habitat thereafter according to the user's will.

At the beginning of the simulation, the user's agent is located at the center of the first room. The user controls it via a motion detection system, which is introduced for facilitating the interaction between the user and the installation in a public exhibition space. In the motion detection window, there are three distinctly colored balls (red, green and blue) that the user can grab and move through gestures (Fig. 3). Among them, the red ball is used to direct the movements of the user's agent. Steering his agent, the user can build new rooms (as soon as a wall is reached) and increase or lower the current room's temperature. The user can use the green ball to control the swarm agents' moving directions or use the blue ball to make the swarm move towards the user's agent.

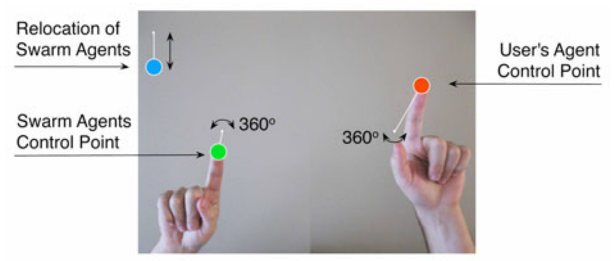

Relocation of Swarm Agents
Swarm Agents Control Point
360°
360°
User's Agent Control Point

Fig. 3. Illustration of what the user sees in the motion detection window. Three colored balls are controlled by the user to direct the agents' movements

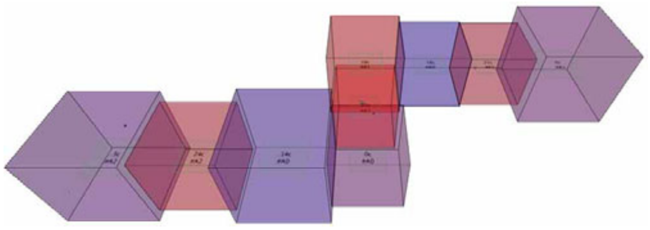


Fig. 4. Continuous space developed in the IBS installation

3.2 The Rooms

The rooms are the building blocks of the architectural design; each room represents a spatial architectural unit that indicates the generation, structure and configuration of the space. Rooms can be assigned with attributes, such as the size and the room temperature.

The three-dimensional rooms are built on the xz-plane. New rooms are always attached to an already existing one, so that a continuous space is created (Fig. 4). Whenever a new room is created, an edge length for the new cube between 50 and 200 pixels is randomly chosen. The temperature in a newly built room is randomly selected from -30 to 30 °C. Rooms with a temperature over 0 °C are displayed in red colors, whereas rooms with a temperature below 0 °C are featured in blue colors. The room temperature can be influenced by both the user's agent and the swarm. In the meantime, different room temperatures trigger various behaviors of the swarm agents (Section 3.3).

3.3 The Swarm Agents

Lastly, the swarm agents function as both the constructors of the architecture and inhabitants of the built space. For instance, they could be seen as both the architects and construction workers of a mall, and the clients and vendors that make use of it. This dual identity is propitious to generate a lively architectural space.

Each time the user builds a room, a new swarm agent, whose gender is randomly assigned, appears at that room center. Without interferences from the user, swarm agents move freely in all directions inside the IBS. The perception field of a swarm agent extends to the whole room in which it is currently located. Its velocity correlates with the perceived room temperature, as explained in more detail in the following paragraphs.

Swarm Agents' Life Cycle. A temperature between 1 and 20 °C is an ideal condition for an agent to reproduce, because it is neither too hot nor too cold. We call a room with this temperature range a delivery room, where male and female agents mate. During pregnancy, the female agent is depicted in grey color instead of white. After 300 frames, a baby agent is given birth to which is represented as a green sphere (Fig. 2). It takes another 300 frames for the baby agent to mature into adulthood. Again, the new agent's gender is randomly assigned. For the parent agents, the male agent has to leave the delivery room at least once before he can mate again; the female agent turns her color back to white after the delivery and is ready for mating again.

Room temperatures not only trigger mating and reproduction but also death. In rooms with temperatures below -20 °C, swarm agents die after 300 frames. We refer to these rooms as death traps.

Altering Local Environments. The swarm agents have the ability to alter their local environments to achieve better living conditions. Firstly, the presence of five agents in any room increases the local temperature by 1 °C. Thus, given a certain number of swarm agents, the extremely low temperature in a death trap can be increased to a level suitable for the swarm to survive. However, this mechanism can also yield extremely high temperatures when large numbers of agents aggregate in one room. To avoid this situation, an additional rule is introduced to limit the individual agent's stay in the warm room (room temperature > 0 °C) to 600 frames. After that period of time, the agent has to leave.

Secondly, when there are 30 or more individuals in one room, the swarm agent that touches a wall first, will try to create a new room. If this new room does not intersect with any neighboring rooms, it will be built (Fig. 5). Consequently, the swarm agents can extend living spaces to alleviate their crowded living condition. The size of the new room is determined by the individual agent who builds it. Generally, the new room size is determined according to the agent gender. A male agent builds a room with a fixed value between 100 and 200 pixels, whereas the female one builds a room with a size between 50 and 100 pixels. The room dimensions of offspring follow their parents' preferences but introduce slight mutations (+/- 10 pixels).

Observed Phenomenon. By coding a set of rules, swarm agents have the tendency to stay at the warm rooms' centers, which makes them more likely to meet others for mating and reproduction. Conversely, swarm agents tend to escape from the center space when the room temperature is below 0 °C, so that they can increase their chances of survival. Therefore, over the course of a simulation, swarm agents have a tendency to congregate in warm rooms.

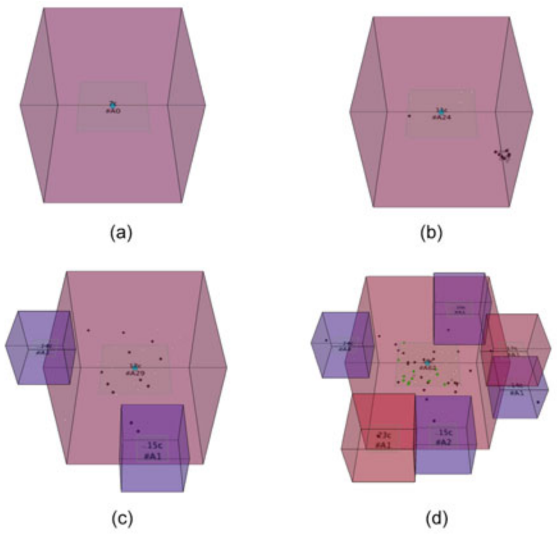


Fig. 5. Swarm agents shaping and exploring their new living space

If it gets too hot (room temperature $> 20\,^\circ$C) in delivery rooms due to overpopulation, those rooms are not used for reproduction any longer. On the other hand, since agents are only allowed to stay in a warm room for at most 600 frames at a time, agents gradually leave those rooms. As the agent numbers are decreased, room temperatures drop and the rooms can be used as delivery rooms again.

Pregnant agents are exempt from the rule of being rushed out of warm rooms after 600 frames. After the delivery, they are subjected to this rule again. Due to this special treatment for pregnant agents, male agents have a greater tendency of ending up in cold rooms. Therefore, it is more likely that male agents become the victims of death traps.

4 Design Results

By specifying a set of interaction rules, the IBS installation informs a self-organizing swarm. In addition, the user is capable of affecting swarm motions whenever it is necessary. During the simulation, both the user's agent and the swarm agents contribute to an architectural design process.

At the very beginning of the simulation, the user typically produces a number of rooms. These rooms are connected and form corridors that serve as the habitat for the swarm agents. Without the support by the swarm agents, the user creates orderly layouts as seen in Fig. 6. As the swarm population grows, the swarm agents begin to explore and extend their living space. The interactions of the self-organizing swarm agents result in interesting, organic-looking designs as shown in Fig. 7.

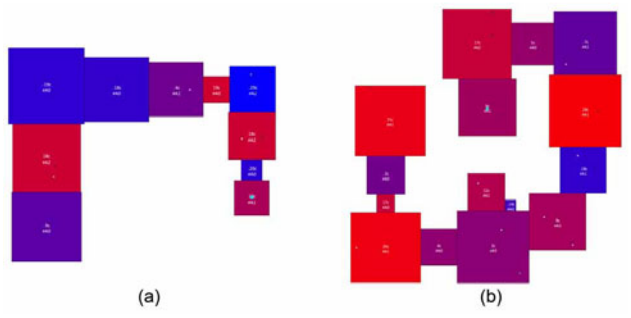


Fig. 6. Two 2D design examples developed by the user

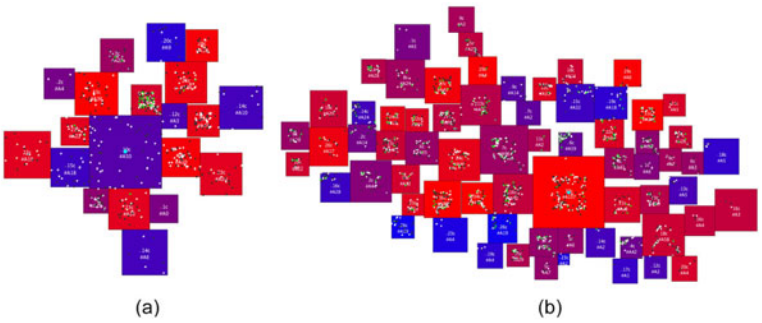


Fig. 7. Two 2D design examples developed mainly by the swarm

The simulation evolves an emotional connection between the user and the swarm. When there are few swarm agents in the system, the user usually feels responsible for populating the space with life. Specifically, when swarm agents went astray into the death traps, the user tends to redirect the agents' movements or tries to change local environmental conditions for the swarm, so that a small quantity of agents can survive. On the other hand, when swarm agents overpopulate the system, the user wants to limit the growth of the swarm agent population. Sometimes, the user might even actively try to reduce the number of agents in the system. Such subtle relationships between the user and the swarm further diversify the design outputs, since the level of the user's intervention determines how much control is relinquished to an external force during the design process.

By combining the efforts by the user and the swarm agents, the design results achieve a balance between systematic, ordered planning and natural, organic growth. Fig. 8 documents such a collaborative work process. First, the user would develop the basic construction layout while swarm agents are busy in expanding their population (Fig. 8(a)). As their population grows to a certain level, swarm agents engage in the construction process (Fig. 8(b) and (c)). As a result, a number of newly built rooms are attached to the old structure (Fig. 8(d)). As time goes on, the original architectural layout by the user is gradually fused

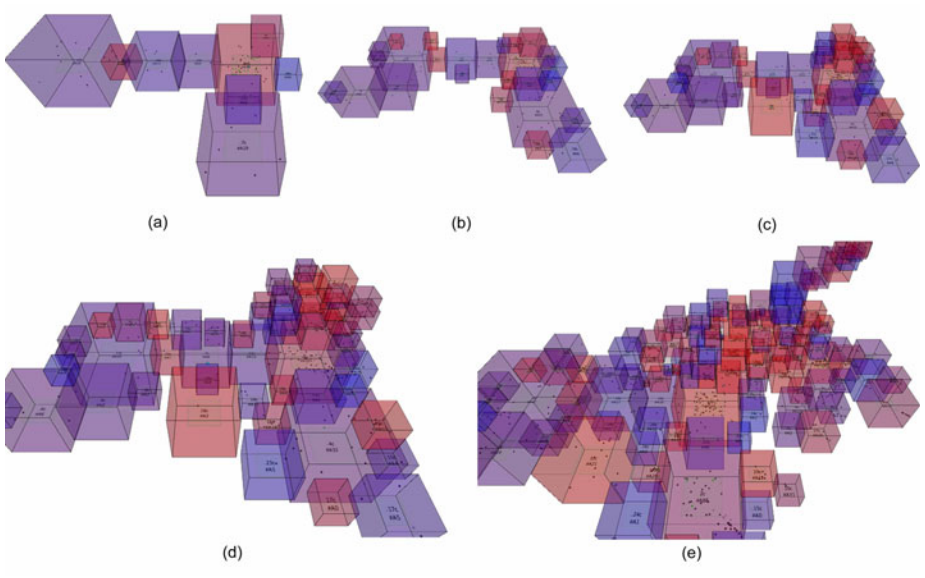


Fig. 8. A collaborative design process between the user and the swarm

and merged into the new settlements. Fig. 8(e) finally exhibits these unfolding settlement patterns. For a personal experience with the IBS installation, please download it at: www.vonmammen.org/ibs/.

5 Future Work

The IBS installation allows a user to create and explore architectural design space. As everywhere on Earth, this space is populated with life, which re-shapes its environment. An interesting, collaborative design process between virtual swarm inhabitants and the user emerges that seeks an ecological balance between order and exploration.

For future work, we consider the following directions to improve the system's usability for architectural design processes. (1) Explore the IBS into the third direction by building new rooms in more than one layer. (2) Diversify the architectural forms by bringing in other building shapes rather than cubes only or even make the agents work on a generic 3D mesh itself by shifting, adding and removing its vertices. (3) Develop sophisticated behavior sets and allow rooms to disappear to achieve a greater structural complexity. (4) Enhance the practicality for architectural modeling by introducing evolutionary algorithms, e.g. by providing separate breeding grounds for individuals swarms and means of interactive evolution to create swarms that support architectural design in unique ways. (5) Install the IBS in a public space and gain user feedback. (6) Translate the built constructions into real-world models and immerse the IBS into a complex virtual space with pre-existing architecture and diverse landscape.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York (1999)
2. Jacob, C.J., Hushlak, G., Boyd, J.E., Nuytten, P., Sayles, M., Pilat, M.: Swarmart: Interactive Art from Swarm Intelligence. Leonardo 40(3), 248–258 (2007)
3. Laland, K., Brown, G.: Niche Construction, Human Behavior, and the Adaptive-Lag Hypothesis. Evolutionary Anthropology 15(3), 95 (2006)
4. McCormack, J., Bown, O.: Life's What You Make: Niche Construction and Evolutionary Art. Applications of Evolutionary Computing, 528–537 (2009)
5. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics 21(4), 25–34 (1987)
6. Unemi, T., Bisig, D.: Music by Interaction among Two Flocking Species and Human. In: Proceedings of the Third International Conference on Generative Systems in Electronic Arts, Melbourne, Australia, pp. 171–179 (2005)
7. von Mammen, S., Jacob, C.: Evolutionary Swarm Design of Architectural Idea Models. In: GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Coputation, pp. 143–150. ACM, New York (2008)

Using Grammatical Evolution to Parameterise Interactive 3D Image Generation

Miguel Nicolau and Dan Costelloe

Natural Computing Research & Applications Group
University College Dublin
Dublin, Ireland
miguel.nicolau@ucd.ie, dan.costelloe@gmail.com

Abstract. This paper describes an Interactive Evolutionary system for generating pleasing 3D images using a combination of Grammatical Evolution and Jenn3d, a freely available visualiser of Cayley graphs of finite Coxeter groups. Using interactive GE with some novel enhancements, the parameter space of the Jenn3d image-generating system is navigated by the user, permitting the creation of realistic, unique and award winning images in just a few generations. One of the evolved images has been selected to illustrate the proceedings of the EvoStar conference in 2011.

1 Introduction

Evolutionary Algorithms permit the creation of candidate solutions from arbitrarily coarse- or fine-grained representations. In an Interactive Evolutionary Design context (i.e. with human evaluators playing the part of the fitness function), user-fatigue is more likely to occur before acceptable quality solutions emerge when the genotype representation is too fine-grained, due to a long, repetitive process with no immediately pleasing results.

For example, consider a toy Evolutionary image generation task where the goal is to produce a simple black and white image on an $N \times N$ grid. The image could be encoded as a binary string of length N^2 – a representation that fits well with a Genetic Algorithm implementation.

This simple representation is powerful in that it permits the construction of *every* possible 2-dimensional monochrome image for a given value of N. If this were a standard, non-interactive optimisation problem, this kind of representation would probably be suitable, since the space of all possible solutions is covered, meaning that with the right conditions, high-quality solutions are almost guaranteed as output from evolution.

But in an interactive setting, this type of (fine-grained) representation makes the construction of even the most basic shapes on the canvas a slow and difficult process. Adding to the difficulty is the potentially destructive nature of the genetic operators of the GA. For this type of problem, the use of pre-defined building blocks (e.g. lines, rectangles, curves) is more likely to produce pleasing or interesting images in a shorter amount of time, while reducing user-fatigue. This notion of building-block creation and re-use has been employed in other artistically focused evolutionary systems such as *Picbreeder* [17].

© Springer-Verlag Berlin Heidelberg 2011

The approach taken in this work was somewhat different than that of typical evolutionary art approaches. In this case, the evolutionary algorithm is not actually constructing the image, but rather parametrising a 3-dimensional visualiser of complex mathematical structures.

There were two main reasons for this approach. The first was speed of development; by using a freely available tool that generates the images, there was no graphical development involved, so all that was required was the integration of the evolutionary algorithm with the visualiser tool, evolving the parameters for the visualiser. The second reason was a shorter evolutionary process. The graphical visualiser used generates fully constructed images, which are quite pleasing to the eye[1]; the time required for the interactive evaluation process to reach images that fulfill the objective is therefore potentially much shorter.

The results obtained are certainly unique; not only that, but they were also obtained with very short runs of the system, thus complying with the objectives stated above. Finally, they are visually appealing, not only for the authors, but also for different communities: one of the evolved images won the EvoStar 2010 art competition, and now illustrates the proceedings cover, while another won a competition to be chosen as the logo for a research cluster, located in the University College Dublin, Ireland.

This paper describes the implementation of this work. It presents the evolutionary algorithm used in Section 2, followed by a brief introduction to Coxeter Matrices and the visualiser used, in Section 3, and finally describes the experiments conducted, in Section 4. Section 5 then draws some conclusions.

2 Grammatical Evolution

Grammatical Evolution (GE) [12,15] is an evolutionary approach that specifies the syntax of possible solutions through a context-free grammar, which is then used to map binary strings onto syntactically correct solutions. Those binary strings can therefore be evolved by any search algorithm; typically, a variable-length genetic algorithm is used.

The use of a grammar to specify the syntax of solutions allows the application of GE to a variety of domains; these are as diverse as horse gait optimisation [8], wall shear stress analysis in grafted arteries [2], and optimisation of controllers for video-games [3]. This also includes earlier applications to evolving art, such as the evolution of logos using Lindenmayer systems [11], musical scores [13], generation of digital surfaces [4], and architectural design [18,7].

2.1 Example Mapping Process

To illustrate the mapping process, consider the (simplified) shape grammar shown in Fig. 1. Given an integer (genotype) string, such as `(4, 5, 4, 6, 7, 5, 9)`,

[1] Although a subjective statement, this opinion was also shared by enough voters to deem one of the images the winner in a competition against other images produced by evolutionary means.

a program (phenotype) can be constructed, which respects the syntax specified in the grammar.

This works by using each integer to choose productions from the grammar. In this example, the first integer chooses one of the two productions of the start symbol `<Pic>`, through the formula $4\%2 = 0$, i.e. the first production is chosen, so the mapping string becomes `<Pic> <Term>`.

The following integer is then used with the first unmapped symbol in the mapping string, so through the formula $5\%2 = 1$ the symbol `<Pic>` is replaced by `<Term>`, and thus the mapping string becomes `<Term><Term>`.

The mapping process continues in this fashion, so in the next step the mapping string becomes `<Var> <Term>` through the formula $4\%2 = 0$, and through $6\%3 = 0$ it becomes `square <Term>`. Continuing in this fashion, all non-terminal symbols in the growing expression are mapped, until the final program becomes `square Rotate(9)`, which can then be used to generate a shape.

```
<Pic>       ::= <Pic> <Term>
              | <Term>
<Term>      ::= <Var>
              | <Op> <Term>
<Op>        ::= Grow(<Value>)
              | Shrink(<Value>)
              | Rotate(<Value>)
<Var>       ::= square
              | circle
              | triangle
<Value>     ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Fig. 1. Example grammar for defining simple shapes

Sometimes the integer string may not have enough values to fully map a syntactic valid program; several options are available, such as reusing the same integers (in a process called wrapping[12]), assigning the individual the worst possible fitness, or replacing it with a legal individual. In this study, an unmapped individual is replaced by its parent.

3 Cayley Graphs

Elementary mathematical group theory teaches us that a group is a special type of set, combined with a number of operations that obey fundamental algebraic rules. For the visually inclined, a Cayley graph permits a diagrammatic representation of the structure of a group with respect to a generating subset. For a given discrete group, G, altering the generating set S can produce visually interesting geometric consequences for the Cayley graph representation of G. An example of this is shown in Fig. 2.

Increasing the complexity of these graphs and corresponding generating sets can in fact generate interesting, visually appealing structures. A complete description of the underlying mathematics at work to create and visualise such

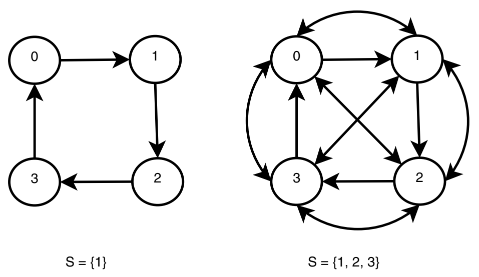


Fig. 2. Two Cayley graphs of the cyclic group $Z/4Z$ produced by two different generating sets

graphs is outside of the scope of this paper, as this would require sections on Coxeter Groups, Reflection Groups, discrete groups, topologies and many other aspects of group theory[2]. There are, however, available tools for the visualisation of such graphs; the Jenn3d system is one of them.

3.1 Jenn3d

Jenn3d [10] is a freely available tool developed by Fritz Obermeyer, which generates visualisations of Cayley graphs of finite Coxeter matrices; it does so using the Todd-Coxeter algorithm, and projects them onto Euclidean 3D space, by embedding them into a 3-sphere.

It is a very simple to use, yet powerful piece of software; its generating parameters are: the Coxeter matrix; the sub-group generators; a set of vertex stabilising generators; specific definitions of edges; specific definitions of faces; and a set of vertex weights.

Thanks to the ability of Grammatical Evolution to evolve parameters to the Jenn3d system that *just work*, it is not necessary to delve into these concepts in any great detail since their complexity can be abstracted away into an image-generating black box. This is obviously very convenient, however we feel that it is also quite a valuable contribution of this work – we as EC researchers and practitioners *do not need to know* the inner complexity of the black box, be it Jenn3d or otherwise. All that is needed is the means to navigate the search space of black box parameters, guiding the system to the generation of visually pleasing images.

4 Experiments

4.1 Setup

A grammar was designed for GE, specifying the exact syntax of the required (and optional) parameters for Jenn3d. Some were tricky to encode; many parameter

[2] Many books are available on the subject, a good resource is Holt *et al* [6].

combinations make Jenn3d crash, or just get stuck in an endless computational loop. The solution to this was to use GE as the main executable, and make external calls to Jenn3d for each evaluation process; if the external call fails, a fitness of 0 (i.e. the worst fitness) is given to that set of parameters.

If the call is successful, the 3D visualiser appears on the screen; the user can then interact with the image, examining its 3-dimensional structure. If a particular viewing angle is found, Jenn3d has options to save a snapshot onto file; additional options were encoded onto Jenn3d for this work: a way to save the parameter combination onto a "best-parameters" file, and a scoring method. Fig. 3 illustrates this.

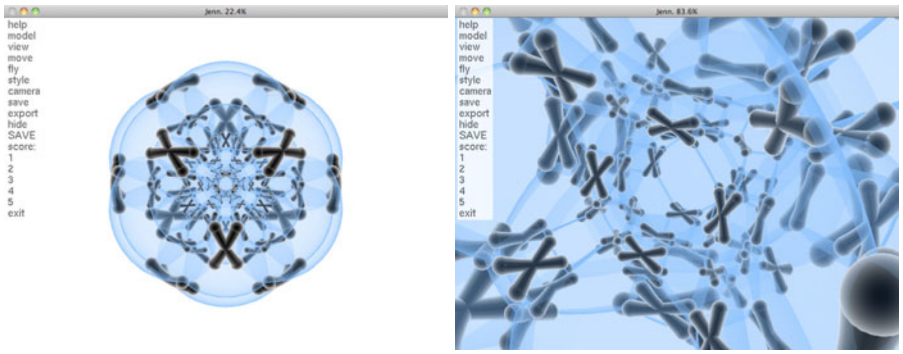


Fig. 3. The Jenn3d interface, along with the extensions encoded. An example image is shown in the left; the same image, when zoomed in and rotated (right), can produce a dramatically different view angle.

The scoring options are the values 1 to 5. If the user gives the score 1 to a structure, it will be replaced by a random structure at the next generation; likewise, due to the 20% elitism replacement used, the structures with the maximum score per generation are guaranteed to remain in the next generation. This ensures that the user retains a degree of control over the generational mechanics, while allowing the evolutionary process to proceed as normal.

Additional novel approaches were used in this work. First, the size of the initial population was larger, to present an initial large spectrum of structures to the user; this size is then culled after the first generation, to the population size chosen for the evolutionary algorithm.

Another novel approach was the encoding of crossover points in the grammar. This is a technique presented recently for GE [9], in which a specific symbol is used in the grammar, to label crossover points; the evolutionary algorithm then only slices an individual according to these points. This technique was particularly useful for the work presented: many of the parameters passed to Jenn3d specify styling options, which can therefore be exchanged as a whole between different individuals (a 2-point crossover operator was used). The same crossover points are used in both individuals, that is, vertex weights will only be exchanged with vertex weights; the length of the exchanged blocks, however, is variable. This makes crossover act mainly as an exploitation operator; as the exchange of parameters can dramatically change the visual appearance of a coxeter matrix, it made sense to limit the role of crossover to the exploration of such combinations. A standard point mutation operation is also used, which ensures the creation of novel parameter values. Fig. 4 shows a section of the used grammar, along with the encoded crossover points.

```
<cmdline> ::= ./jenn <GEXOMarker> -c <CoxeterMatrix> <GEXOMarker>
           <StabilizingGenerators> <GEXOMarker> <Edges> <GEXOMarker>
           <Faces> <GEXOMarker> <VertexWeights> <GEXOMarker>
<CoxeterMatrix> ::= <Torus> | <FreePolyhedra> | <FreePolytope>
<StabilizingGenerators> ::= "" | -v <Comb0123>
<Edges>         ::= "" | -e <EdgeSet>
<Faces>         ::= "" | -f <FaceSet>
<VertexWeights> ::= "" | -w <Int1-12> <Int1-12> <Int1-12> <Int1-12>
```

Fig. 4. Section of the grammar used; crossover points are encoded using the special non-terminal symbol `<GEXOMarker>`

The experimental parameters used are shown in Table 1. To ensure all individuals in the initial population were valid, a form of population initialisation [14] was used. Also, the mutation rate was set such that, on average, one mutation event occurs per individual (its probability is dependent on the length of each individual). Finally, note that there is no maximum number of generations; evolution will always continue, until the user decides to terminate the execution.

Table 1. Experimental Setup

Initial Population Size	20
Evolutionary Population Size	10
Derivation-tree Depth (for initialisation)	10
Selection Tournament Size	10%
Elitism (for generational replacement)	20%
Crossover Ratio	50%
Average Mutation Events per Individual	1

4.2 Results

A typical run of the system seems to give many undesirable images on the first generation; some cause Jenn3d to crash, while others are visually displeasing. After the first generation, however, the system seems to settle onto a sequence of very pleasing images, based on variations of the initial best images.

There is always novelty being introduced into the population, and the user has an active part on this process, by attributing a fitness score of 1 to displeasing

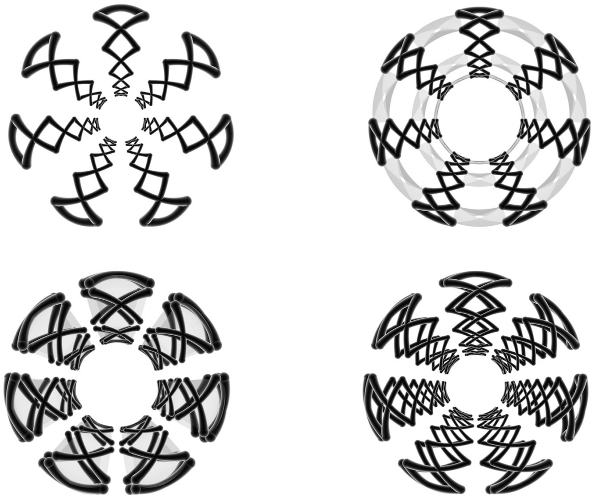


Fig. 5. Example image and close variations achievable with the genetic operators used

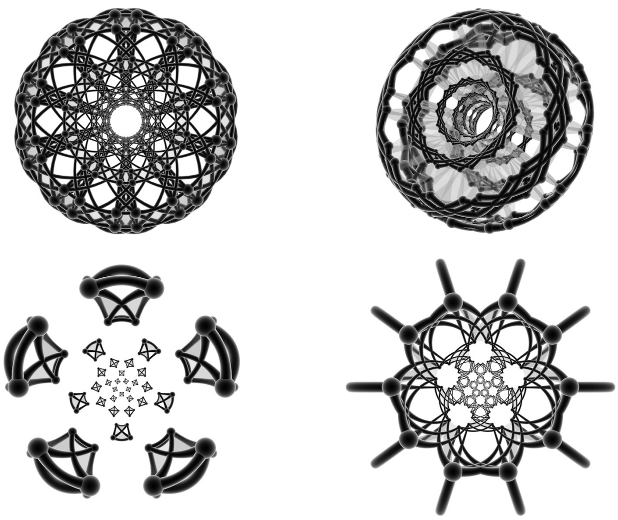


Fig. 6. Example of the variety of structures achievable from a single run

structures, which forces these to be replaced by novel ones. A single run of the system can therefore generate a wide variety of images; once a user has explored many variations of a style, he/she can start attributing them fitness scores of 1, which effectively purges the population of these structures, and ensures that new, unseen structures are present in the next generation.

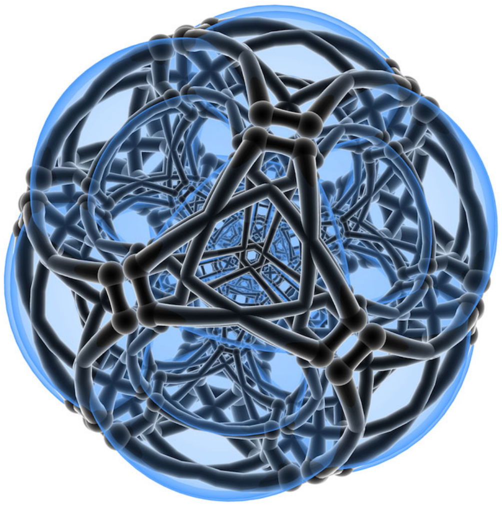


Fig. 7. The winning image, elected to illustrate the 2011 EvoStar proceedings

Fig. 5 shows examples of different variations of a pleasing image, obtained mostly through the exploration power of the new crossover operator. Fig. 6, on the other hand, shows many different images extracted from a single run of the system, a result achievable through the user-control technique explained. This is possible both due to the variety of structures which Jenn3d can project, and also to the exploration of the parameter space by GE. Note that all these figures are shown zoomed out; rotation, zooming and *fly-in* transformations can seriously alter the style achieved. Finally, Fig. 7 shows the image that won the EvoStar 2010 Art Competition.

5 Conclusions

This paper has presented a novel application of Grammatical Evolution to Evolutionary Art that treats the task of producing pleasing Jenn3d / Coxeter visualisations as a parameter search problem. In fact, GE has been particularly well suited to this problem thanks to the construction of a grammar that encapsulates the complexity of the parameters of the image-generating Jenn3d system. The ease of use of both GE and Jenn3d made them easily combinable, which not only resulted in a fast implementation, but also allowed the generation of unique and very often pleasing images.

A fair criticism of the use of black-box such as Jenn3d is that the images produced will always be limited to the space of possibilities that the black-box is capable of creating. This is indisputable – doing so constrains the space of potential solutions and for certain problems this should be avoided in order to gain maximum coverage of the search space.

However, there are also cases where jump-starting is a sensible thing to do. In this case, the space of possible solutions is still sufficiently large that a spread of pleasing and not so pleasing images can come about. The necessity of interactive fitness assignment is just as much a requirement as it would be for a system producing arbitrary images. The advantage of using a system such as Jenn3d is that the user will not spend time in the early generations evolving the fundamentals.

The encoding of crossover points in the grammar also worked with great effect in this work. The crossover operator was originally designed [5] to work just like in nature, that is, to allow two chromosomes to exchange building-blocks; there has been a great dispute over the years, however, as to the real exploitation nature of crossover, and in fact to the existence of exchangeable building-blocks in Genetic Programming systems [1,16]. In this work, they do exist, and the crossover operator was encoded to take full advantage of this fact.

Finally, some of the images generated by this approach were submitted to a few competitions, with award-winning results: one has won the Evolutionary Art competition at EvoStar 2010, and another has been chosen as a logo representation for a research cluster in University College Dublin.

References

1. Angeline, P.J.: Subtree crossover: Building block engine or macromutation? In: Koza, J.R., et al. (eds.) Proceedings of Genetic Programming 1997: Second Annual Conference, Stanford, USA, July 13-16 (1997)
2. Azad, R.M.A., Ansari, A.R., Ryan, C., Walsh, M., McGloughlin, T.: An evolutionary approach to wall shear stress prediction in a grafted artery. Applied Soft Computing 4(2), 139–148 (2004)
3. Galván-López, E., Swafford, J.M., O'Neill, M., Brabazon, A.: Evolving a ms. pacman controller using grammatical evolution. In: Di Chio, C., et al. (eds.) EvoApplicatons 2010. LNCS, vol. 6024, pp. 161–170. Springer, Heidelberg (2010)
4. Hemberg, M., O'Reilly, U.M.: Extending grammatical evolution to evolve digital surfaces with genr8. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 299–308. Springer, Heidelberg (2004)
5. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
6. Holt, D.F., Eick, B., O'Brien, E.A.: Handbook of Computational Group Theory (Discrete Mathematics and Its Applications). Chapman and Hall/CRC, Boca Raton (2005)
7. McDermott, J., Griffith, N., O'Neill, M.: Interactive EC control of synthesized timbre. Evolutionary Computation 18(2), 277–303 (2010)
8. Murphy, J.E., O'Neill, M., Carr, H.: Exploring grammatical evolution for horse gait optimisation. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 579–584. Springer, Heidelberg (2009)
9. Nicolau, M., Dempsey, I.: Introducing grammar based extensions for grammatical evolution. In: Proceedings of IEEE Congress on Evolutionary Computation, CEC 2006, Vancouver, BC, Canada, July 16-21, pp. 2663–2670. IEEE Press, Los Alamitos (2006)
10. Obermeyer, F.: Jenn3d for visualizing coxeter polytopes (June 2010), http://www.math.cmu.edu/~fho/jenn/

11. O'Neill, M., Brabazon, A.: Evolving a logo design using lindenmayer systems, postscript & grammatical evolution. In: Proceedings of IEEE Congress on Evolutionary Computation, CEC 2008, Hong-Kong, June 1-6, pp. 3788–3794. IEEE Press, Los Alamitos (2008)
12. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, Genetic programming, vol. 4. Kluwer Academic Publishers, Dordrecht (2003)
13. Reddin, J., McDermott, J., Brabazon, A., O'Neill, M.: Elevated pitch: Automated grammatical evolution of short compositions. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 579–584. Springer, Heidelberg (2009)
14. Ryan, C., Azad, R.M.A.: Sensible initialisation in grammatical evolution. In: Barry, A.M. (ed.) GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, pp. 142–145. AAAI, Chigaco (2003)
15. Ryan, C., Collins, J.J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) First European Workshop on Genetic Programming 1998, pp. 83–95. Springer, Berlin (1998)
16. Sastry, K., O'Reilly, U.M., Goldberg, D.E., Hill, D.: Building block supply in genetic programming. In: Riolo, R., Worzel, B. (eds.) Genetic Programming Theory and Practice, ch. 4, pp. 137–154. Kluwer Publishers, Boston (2003)
17. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Stanley, K.O.: Picbreeder: evolving pictures collaboratively online. In: Proceeding of the Twenty-sixth Annual SIGCHI Conference on Human Factors in Computing Systems, CHI 2008, pp. 1759–1768. ACM, New York (2008)
18. Shao, J., McDermott, J., O'Neill, M., Brabazon, A.: Jive: A generative, interactive, virtual, evolutionary music system. In: Di Chio, C., et al. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 341–350. Springer, Heidelberg (2010)

Evolving Textures from High Level Descriptions: Gray with an Accent Color

Craig Reynolds

Sony Computer Entertainment, US R&D
craig_reynolds@playstation.sony.com

Abstract. This paper describes a prototype evolutionary texture synthesis tool meant to assist a designer or artist by automatically discovering many candidate textures that fit a given stylistic description. The textures used here are small color images, created by procedural texture synthesis. This prototype uses a single stylistic description: a textured gray image with a small amount of color accent. A hand-written prototype fitness function rates how well an image meets this description. Genetic programming uses the fitness function to evolve programs written in a texture synthesis language. A tool like this can automatically generate a catalog of variations on the given theme. A designer could then scan through these to pick out those that seem aesthetically interesting. Their procedural "genetic" representation would allow them to be further adjusted by interactive evolution. It also allows re-rendering them at arbitrary resolutions and provides a way to store them in a highly compressed form allowing lossless reconstruction.

Keywords: texture synthesis, evolutionary computation, genetic programming, GP, evolutionary art, design, tool.

1 Introduction

Many aspects of visual art and design make use of texture patterns. Textures are a basic component of 2D graphic design, such as for web sites or printed material, as well as 3D graphic design, such as for 3D animation, realistic effects for movies, and real time 3D in virtual worlds and games. In 3D applications, 2D textures can be mapped onto 3D surfaces, and put to other uses in shading and texture-based modeling.

These textures are sometimes photographic: landscapes and clouds for backgrounds, closeup photography for woodgrains or cloth textures. Other times the required texture is more generic and may be specified in terms of brightness, color or pattern. This paper describes a prototype tool for generating textures to match a certain specification. Large collections of such textures can be made automatically then presented to the artist or designer to be considered as candidates for the intended purpose.

In these experiments a single texture "style" is used. The fitness function that defines the style was written by hand in C++. Finding a way an artist could create these specifications for themselves is a key topic for future work.

© Springer-Verlag Berlin Heidelberg 2011

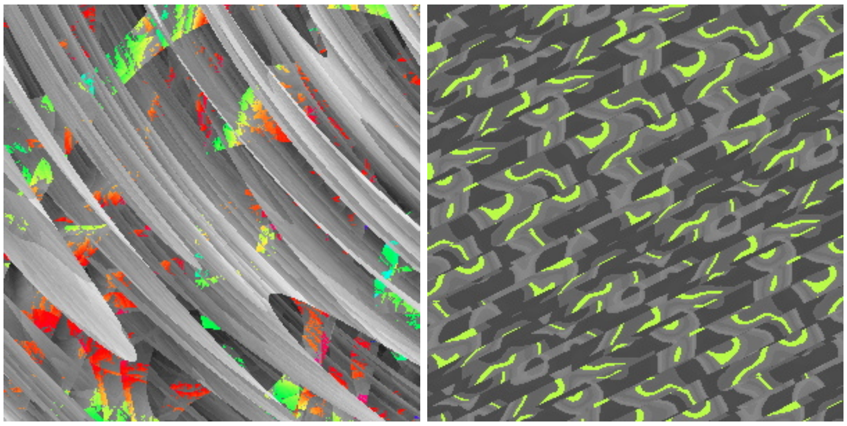


Fig. 1. Two textures hand-selected from several hundred automatically generated textures meeting a high-level stylistic description: a textured gray image with a small amount of color accent. Typical runs took between 1 and 5 minutes to complete. (Color images and other information available at `http://www.red3d.com/cwr/gots/`).

2 Related Work

Interactive evolution of images was first demonstrated in software that accompanied Richard Dawkins' 1986 book *Blind Watchmaker* [2]. In 1991 Karl Sims expanded the idea and applied it to procedural synthesis of color textures [14]. Since then quite a bit of work has appeared, notably including *Electric Sheep* [6], an animated and crowd-sourced version by Scott Draves. Ken Stanley and colleagues have described texture synthesis using evolved neural nets [15].

Less well studied is "non-interactive" evolution of images in a traditional evolutionary computation (run unsupervised, using a procedural fitness function). Visual art is easily evaluated by the human visual system in interactive evolution, while in unsupervised evolution, the complex tasks of visual understanding and evaluation must be encoded into software. Neither vision nor aesthetics are easily simulated, making the evolution of visual art an unsolved if not ill-posed problem. Nonetheless attempts have been made to explore this topic, as recently surveyed by den Heijer and Aiben [3,4]. DiPaola and Gabora [5] attempt to model human creativity. See also a thorough critique of the field by Galanter [8].

A more tractable related problem is to evolve a synthetic image to minimize its difference from a given target image. Small differences sometimes provide visually interesting "stylistic" variations on the target image. Examples of this approach are seen in Wiens and Ross' *Gentropy* [16], and in work by Alsing [1]. Hertzmann used a similar approach based on relaxation [9].

The approach taken in this work is even more tractable: posing a relatively "easy" problem for evolution to solve, collecting batches of solutions, then using human visual judgement to cull the interesting results from the mundane. An unresolved issue is whether this relatively simple evolutionary computation will actually provide useful help to a working designer. This work uses a slightly different approach to evolutionary texture synthesis. In most previous work, evolved textures were represented by programs that compute an individual pixel's color, given its coordinates. The GP function set used here is based on passing objects that represent entire images. This approach to evolutionary texture synthesis [12] has been used to model the evolution of camouflage in nature [13].

3 Implementation

The evolutionary procedural texture synthesis used in this work is based on three components: an engine for running genetic programming, a GP function set drawn from a library of texture generators and operators [12], and a fitness function. The GP engine is provided by *Open BEAGLE* [7,11] an excellent general purpose toolkit for evolutionary computation.

The GP function set includes about 50 functions from the texture synthesis library [12]. These functions return an object of type *Texture* and may also take textures as input. The grammar also includes types for 2d Cartesian vectors, RGB colors and five numeric types differentiated by range (for example, fractions on [0, 1], small signed values, etc.). Open BEAGLE's built in support for Montana's *strongly-typed genetic programming* [10] accommodates this mixture of types. The GP population is 100, divided into 5 demes of 20 individuals each. Several of Open BEAGLE's genetic operators are used, but primarily evolution is based on GP crossover and "jiggle" mutation of ranged floating point constants. The termination condition for these runs is 50 generations (hence 5000 fitness tests) or when an individual reached 95% fitness. Textures shown here are rendered at 300x300 pixels, fitness evaluation was done at 100x100 resolution. See Fig. 7 for an example of texture and source code.

4 Fitness Function for a Graphical Style

At the center of this work is a handwritten fitness function used to score each texture on how well it meets the prototype criteria of "a textured gray image with a small amount of color accent." The fitness function needs to take a procedural texture object and returns a numerical fitness on the range [0, 1]. The evolutionary computation seeks to maximize this value, so 1 is perfect fitness.

The approach taken here is to establish several independent criteria, based on various properties of the image. These can be based on isolated pixels (brightness) or on larger neighborhoods (spatial frequencies). In these experiments there were five independent criteria, suggesting that this would require a *multi-objective evolutionary algorithm.* There are well established MOEA techniques, some of which are provided with Open BEAGLE.

Instead this work used a simple transformation from multi-objective to single objective. If each independent criteria is scored from 0 to 1, the scores can simply be multiplied together to form a *product of fractions.* This does not allow moving independently on fitness hyperplanes as in Pareto optimization. It does have two

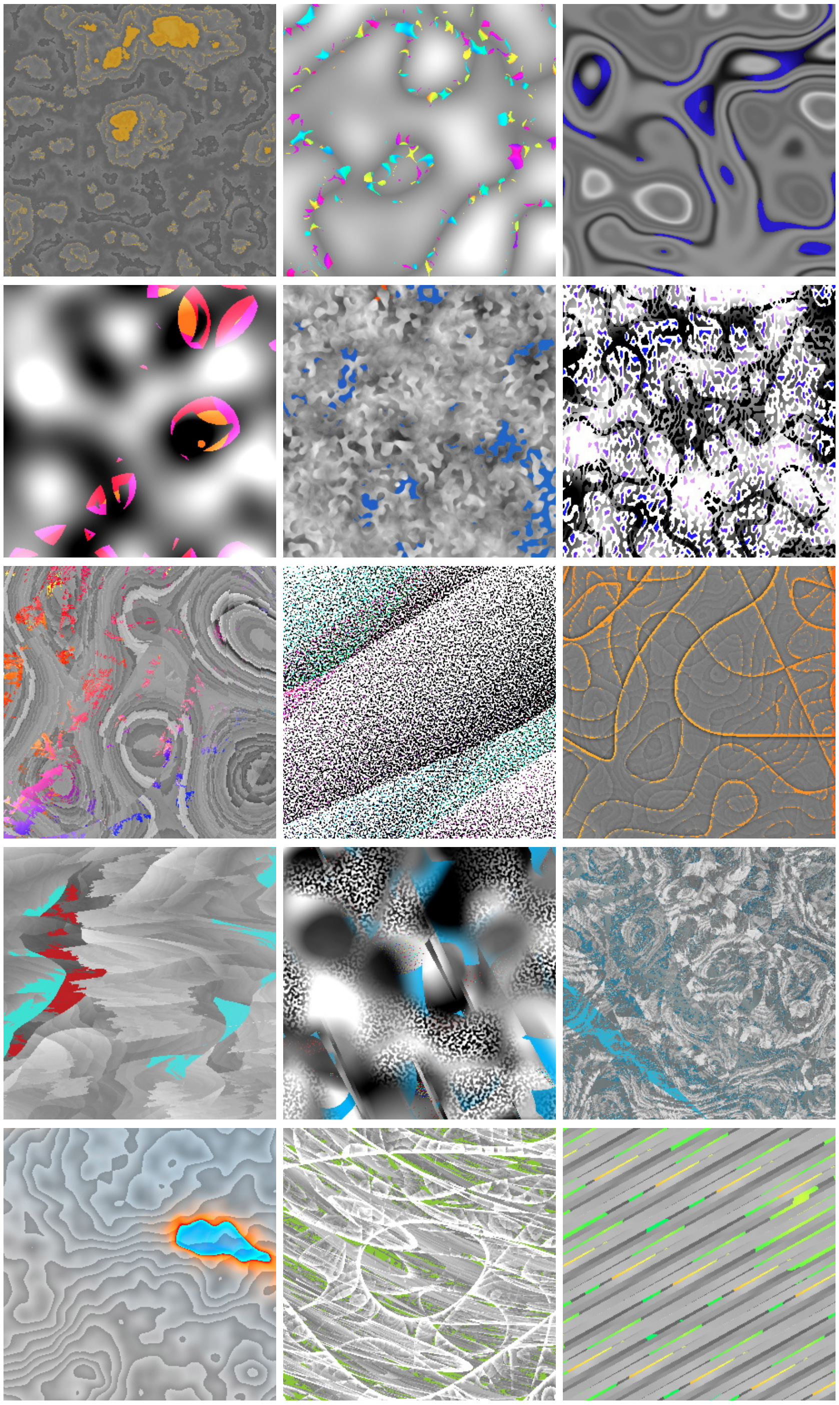


Fig. 2. More examples of textures evolved to meet the same stylistic criteria

useful properties: a change in any single score adjusts the final fitness in the same direction, and any low score acts as a fitness “veto” on other high scores. This simplified approach to multi-objective fitness seemed to work well enough for these experiments. (Based on a small test: combining scores as a renormalized sum (average) seems to result in slower and less effective evolution.)

The fitness function looks for a mostly gray image with a certain amount of bright colors, it prefers brightness fluctuations, and it slightly prefers midrange images. It samples the texture and counts gray pixels (saturation below a given threshold (0.2)) and colored pixels (above a second saturation threshold (0.6)). It also measures how close each pixel’s brightness is to middle gray, computes the RGB bounding box of all colored pixels, and a measure of variability over small random neighborhoods. From these statistics five fitness criteria are computed:

1. fraction of good pixels (below gray threshold or above color threshold)
2. how close ratio of color/good is to given target value (0.05)
3. average score for “midrangeness”
4. fraction of variability samples above a given contrast threshold (Fig. 6)
5. size of the colored pixel bounding box (to promote non-uniform colors)

These values were adjusted to be above 0.01 to retain fitness variation even for the very low values typical at the beginning of a run. Some tuning was used, for example to reduce the strength of the “midrangeness” criteria (number 3). Its value was remapped onto the more gentle forgiving range [0.9, 1.0], so an average brightness of 50% gave a score of 1.0 while even a very dark texture would get only a slightly lower score of 0.9. Typical results are shown in Fig. 1 and Fig. 2. Some runs were tried with “midrangeness” set to favor bright or dark textures, see Fig. 3 and Fig. 4.

5 Results

It proved relatively easy to find textures that met the fitness threshold of 0.95 (95%) within 50 generations. Most runs did so in 10-30 generations. In the envisioned application of this technique, evolving an acceptable texture is only the beginning of the process. After many such textures are found, a human observer inspects them and picks out a few that seem especially interesting. (See more information at `http://www.red3d.com/cwr/gots/`)

Each evolution run produces one texture. This process generally took between one and five minutes to complete (on a MacBook Pro, 2.8 GHz Intel Core 2 Duo). Some runs took only took a few seconds, some took more than ten minutes. Occasionally some runs took longer due to very expensive evolved programs (with many calls to convolution-based operators) and perhaps due to some computational inefficiencies in Open BEAGLE’s STGP crossover operator.

Many of the results from these experiments were visually similar. Certain themes seemed to reappear frequently. Most common in early runs were textures composed of a gray pattern with its darkest (or brightest) parts replaced with a uniform color. Sometimes the gray portion was simply one of the monochrome

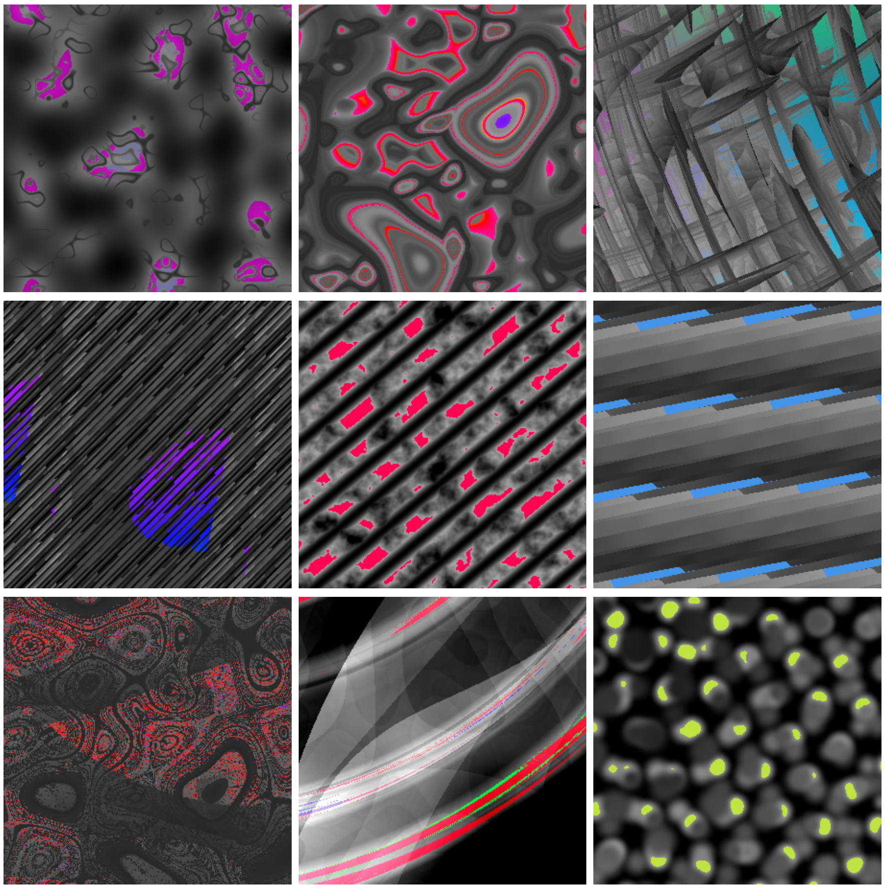


Fig. 3. Dark textures evolved with target brightness set low (or disabled)

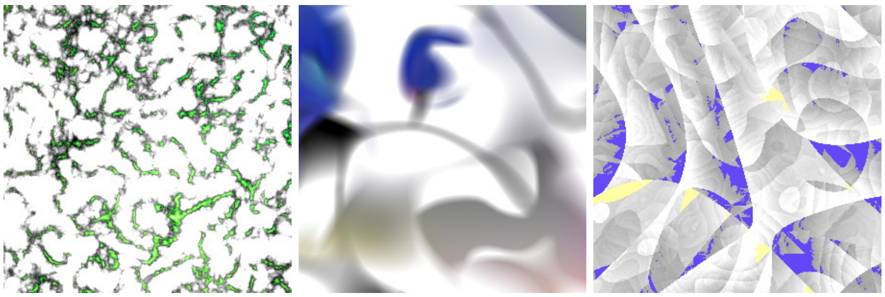


Fig. 4. Bright textures evolved with target brightness set high (or disabled)

texture generators from the texture synthesis library, leading to uninteresting simplistic solutions to the fitness function such as: *Max (Noise (...), UniformColor (...))* Trying to avoid these, some runs were made with *Max* and *Min* removed

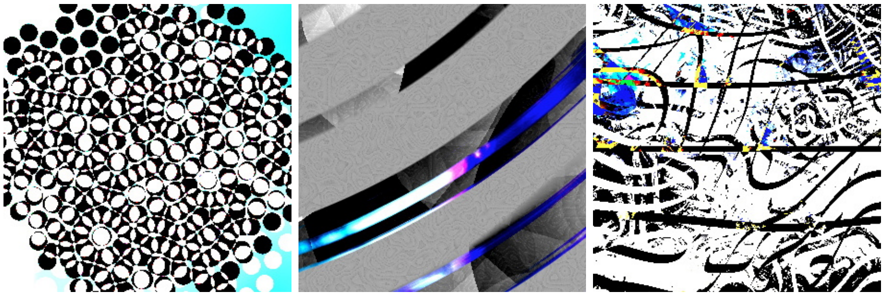


Fig. 5. Textures evolved with *Max* and *Min* removed from GP function set

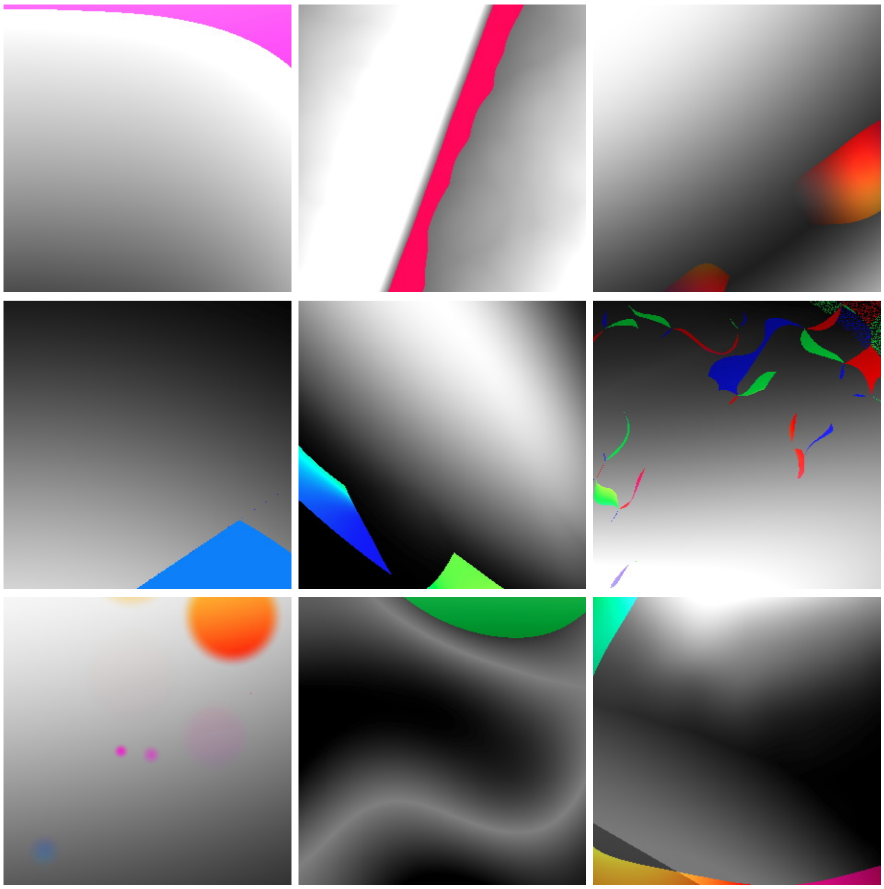


Fig. 6. Several accidentally "minimalist" textures evolved during these experiments. Textures with large areas of constant brightness were selected against. Textures shown here were near the lower limit for acceptable rates of variation.

from the GP function set. While this seemed to make the problem harder for evolution to solve (fewer successful runs) it produced some novel kinds of textures, see Fig. 5. In later runs a fitness metric was added to reward textures with a range of colors.

Many successful runs produced these trivial solutions with a pre-defined gray texture and small irregular splotches of color. These trivial solutions often corresponded with short runs. If a solution was found in the first 10 generations it was often one of these technically simplistic, visually boring textures. Long runs often produced more interesting textures, but sometimes a long run produced one of the common, boring textures. Runs that ended with less than 95% fitness necessarily completed all 50 generations, and often did not appear to fit the definition of “a textured gray image with a small amount of color accent”.

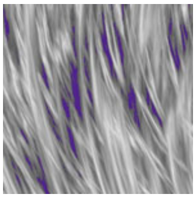
6 Conclusions and Future Work

The results of this experiment are encouraging. At a superficial level, on this one prototype application, this technique of evolving textures to meet a high level design goal seems to work. At least it seems able to easily generate a large range of candidate textures which can then be culled by hand to produce a few novel, unexpected and visually interesting textures.

As mentioned above, many textures generated with this technique had a certain sameness. This is not necessarily a problem, the human eye is quite good at scanning over many similar items while looking for a special one. However the similarity of many results suggests that the random sampling of texture synthesis space is happening in a non-uniform manner. There appear to be several large *basins of attraction* in this space. Once a search wanders into such a basin it is likely to produce a certain type of result. Finding a way to shrink these basins, or to limit the samples allocated to them, might allow the space of textures to be explored more efficiently. One possible approach would be to let the human user occasionally inspect the evolving population and cull out the overrepresented or uninteresting textures. This kind of negative selection would be similar to the “human predator” in a recent hybrid model of camouflage evolution [13].

In this work the only goal was to generate visually interesting textures. In a real application an artist would be trying to solve an artistic design problem and that would impose additional constraints on the generated textures, effectively reducing the “yield” of interesting textures. It remains to be seen if this technique could actually provide useful help to an artist in their design work. Sometimes there are very tight constraints on what textures are appropriate for a certain use. Other times the criteria are very broad (“find any dark fine-grained brownish texture”). At least in the latter case, this technique could be quite helpful.

A useful property of procedural texture synthesis is that the “source code” can serve as a highly compressed representation of the texture. The textures used in this work use floating point pixel coordinates and so can be rendered at any required resolution. Together these mean that a multi-megapixel texture can be stored as several hundred characters. Source code for textures can also be put into a Sims-like interactive evolution tool [14] to allow artist-guided refinement.



```
Max (Max (UniformColor (Pixel (0.308089, 0.127216, 0.564523)),
          VortexSpot (1.23485, 5.30871, Vec2 (1.99217, 0.137068),
                      Furbulence (0.152681, Vec2 (-1.74168, 0.119476)))),
     VortexSpot (1.23485, 5.30871, Vec2 (2.91655, 0.119476),
                 VortexSpot (1.23485, 5.30871, Vec2 (1.99217, 0.138486),
                             Max (UniformColor (Pixel (0.308089, 0.127216, 0.564523)),
                                  Furbulence (0.35606, Vec2 (2.91655, 0.119476))))))
```

Fig. 7. A "wispy" high-fitness (99.98%) texture and its concise evolved source code

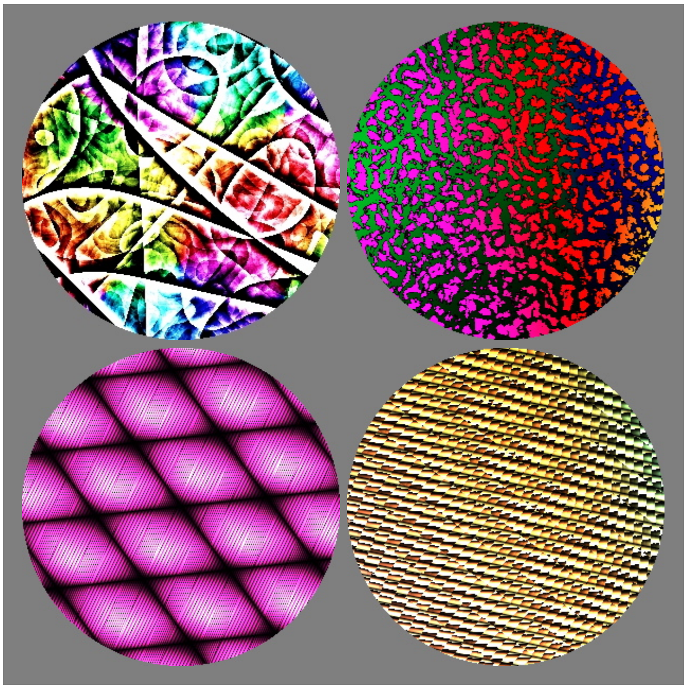


Fig. 8. Textures evolved using another fitness function, similar in structure to the "gray with accent color" style but instead looking for textures with high frequency variation, a significant amount of color and a brightness histogram that is relatively flat [12]

The larger question about the utility of this technique is how fitness functions will be constructed for arbitrary design criteria. In the single case considered here, it was programmed in C++ over the course of about one day, interspersed with many trial evolution runs. More experience is needed to evaluate how much effort is required to write fitness functions for other kinds of design criteria. (At least one other prototype example exists, see Fig. 8 and [12]) Also of interest is the degree to which these functions have certain regularities and modularities. If so it would allow creating a library of texture evaluation tools to help program these fitness functions. It might also lead to interactive tools allowing artists to construct their own evaluation criteria for texture evolution without requiring programming skills. Ultimately, putting texture synthesis in competition with texture analysis could lead to interesting new coevolutionary systems.

References

1. Alsing, R.: Genetic Programming: Evolution of MonaLisa (2008), `http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/`
2. Dawkins, R.: The Blind Watchmaker. W. W. Norton, New York (1986)
3. den Heijer, E., Eiben, A.E.: Using Aesthetic Measures to Evolve Art. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 321–330. Springer, Heidelberg (2010)
4. den Heijer, E., Eiben, A.E.: Comparing aesthetic measures for evolutionary art. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 311–320. Springer, Heidelberg (2010)
5. DiPaola, S., Gabora, L.: Incorporating characteristics of human creativity into an evolutionary art algorithm. Genetic Programming and Evolvable Machines 10(2), 97–110 (2009)
6. Draves, S.: The Electric Sheep and their Dreams in High Fidelity. In: Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering (NPAR 2006), pp. 7–9. ACM, New York (2006), `http://electricsheep.org/`
7. Gagné, C., Parizeau, M.: Genericity in Evolutionary Computation Software Tools: Principles and Case-Study. International Journal on Artificial Intelligence Tools 15(2), 173–194 (2006)
8. Galanter, P.: The Problem with Evolutionary Art Is... In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 321–330. Springer, Heidelberg (2010)
9. Hertzmann, A.: Paint By Relaxation. In: Proceedings of the Computer Graphics International Conference, pp. 47–55. IEEE Computer Society, Los Alamitos (2001)
10. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1995)
11. Open BEAGLE, `http://beagle.gel.ulaval.ca/`
12. Reynolds, C.: Texture Synthesis Diary (2010), `http://www.red3d.com/cwr/texsyn/diary.html`
13. Reynolds, C.: Interactive Evolution of Camouflage. In: Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems (ALife XII). MIT Press, Cambridge (2010), `http://www.red3d.com/cwr/iec/`
14. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of SIGGRAPH 1991, pp. 319–328. ACM, New York (1991)
15. Stanley, K.: Compositional Pattern Producing Networks: A Novel Abstraction of Development. Genetic Programming and Evolvable Machines 8(2), 131–162 (2007)
16. Wiens, A.L., Ross, B.J.: Gentropy: Evolutionary 2D Texture Generation. Computers and Graphics Journal 26, 75–88 (2002)

Aesthetic Classification and Sorting Based on Image Compression

Juan Romero[1], Penousal Machado[2], Adrian Carballal[1], and Olga Osorio[3]

[1] Faculty of Computer Science, University of A Coruña, Coruña, Spain
jj@udc.es, adriancarballal@gmail.com
[2] CISUC, Department of Informatics Engineering,
University of Coimbra, 3030 Coimbra, Portugal
machado@dei.uc.pt
[3] Faculty of Communication Sciences, University of A Coruña, Coruña, Spain
olga.osorio@udc.es

Abstract. One of the problems in evolutionary art is the lack of robust fitness functions. This work explores the use of image compression estimates to predict the aesthetic merit of images. The metrics proposed estimate the complexity of an image by means of JPEG and Fractal compression. The success rate achieved is 72.43% in aesthetic classification tasks of a problem belonging to the state of the art. Finally, the behavior of the system is shown in an image sorting task based on aesthetic criteria.

1 Introduction

Having an estimate of aesthetic value, allowing the differentiation among various objects based on merely aesthetic criteria, would have a great theoretical and practical value in the field of Evolutionary Art.

This paper presents a set of 18 features, based on JPEG and Fractal compression, paying attention to the complexity of an image. Their adequacy is shown in two different aesthetic tasks: classification and sorting. First of all, we tackle the issue of image classification based on aesthetic criteria presented by Datta et al. [4]. Using both the image dataset and the features provided by them, a thorough comparison was established with those detailed in the present paper by means of Support Vector Machines (SVMs) and ANNs. A linear combination of the outputs of the neural network trained in the previous task is used to sort several image sets presented by [10]. That combination is presented as a possible aesthetic fitness function which we intend to use within an Evolutionary Art System in the future.

2 Complexity and Aesthetics

The relationship between aesthetics and image complexity has been explored in several psychology and graphic computation papers [2,5,6,16]. In a simplified

© Springer-Verlag Berlin Heidelberg 2011

way, the complexity of an image is related to its entropy, and inversely related to the order. It is related to the minimal information (or the minimal program) required to "construct" the image. It may be said to depend on the degree of predictability of each pixel of the image [17]. Thus, a flat image with every pixel of the same color shows a perfect order, and it is less complex. A pure random image can be seen as extremely complex and the value of each pixel is impossible to predict, even taking into account the values of neighbor pixels.

The relevance of perceived image complexity is a recurring topic in the field of aesthetics [1,2,17]. According to [12], "Aesthetic value is related to the sensorial and intellectual pleasure resulting from finding a compact percept (internal representation) of a complex visual stimulus". In the same paper, two different estimates are presented: one for the Complexity of the Visual Stimulus (CV), using JPEG Compression and another for the Complexity of the Percept (CP), using Fractal Compression. Finally, the metrics are tested with psychological test: "Design Judgment Test" [8]. In [15], Machado used a subset of the features proposed in this project and an Artificial Neural Network (ANN) classifier for author identification, attaining identification rates higher than 90% across experiments. This paper presents an aesthetic fitness function based on the metrics proposed by [12].

3 Proposed Features

While several preceding works [4,10,20] use ad-hoc metrics designed for a specific problem, the present paper will use general metrics based on edge detection and complexity estimates of black and white images. The said estimates are determined from the compression error generated from the original image. The advantage posed by these metrics is their generality; they are easily estimated and can be applied only on grayscale information of the image.

Before carrying out the calculations of the different features, every image is individually subjected to a series of transformations before being analyzed. A given input image is loaded and resized to a standard width and height of 256×256 pixels, transformed into a three channel image in the RGB (red, green and blue) color space, with a depth of 8-bit per channel and all pixel values scaled to the $[0, 255]$ interval. This step ensures that all input images share the same format and dimensions.

Afterwards, the image is converted into the HSV (Hue, Saturation and Value) color space and its HSV channels are split. Only the V channel is stored as a 1-channel grayscale image, given that we just need its representation in black and white format.

Previous works such as [4,10,11] rely, to a large extent, on color information to extract features. [10] states "the color palette seen in professional photos and snapshots is likely to be very different". In this work, we rely exclusively on grayscale information. We want to make the system as generic as possible, and in every dataset we have there are some grayscale images. In the future, however, we will analyze the results by using also color information (channels HS).

Once the grayscale image is available, two edge detection filters are applied, Canny and Sobel, which will yield two new black and white images. In previous works (e.g., [18,10]) filters such as Canny, Sobel, Gauss and Laplace have been applied.

The most popular image compression schemes are lossy, therefore they yield a compression error, i.e., the compressed image will not exactly match the original. All other factors being equal, complex images will tend towards higher compression errors and simple images will tend towards lower compression errors. Additionally, complex images will tend to generate larger files than simple ones. Thus, the *compression error* and *file size* are positively correlated with image complexity [9]. To explore these aspects, we consider three levels of detail for the JPEG and Fractal compression metrics: *low*, *medium*, and *high*. The process is the same for each compression level; the current image in analysis is encoded in a JPEG or fractal format. We estimate each metric of image I using the following formula:

$$RMSE(I, CT(I)) \times \frac{s(CT(I))}{s(I)} \tag{1}$$

where $RMSE$ stand for the root mean square error, CT is the JPEG or fractal compression transformation, and s is the file size function.

In the experiments described herewith, we use a quad-tree fractal image compression scheme [7] with the set of parameters given in Table 1. Note that letting the minimum partition level be 3 implies that the selected region is always partitioned into 64 blocks first. Subsequently, at each step, for each block, if one finds a transformation that gives good enough pixel by pixel matches, then that transformation is stored and the image block isn't further partitioned. (Here, pixel by pixel match is with respect to the usual 0 to 255 grayscale interval encoding.) If the pixel by pixel match error is more than 8 for at least one of the pixels of the block in the partition, that image block is further partitioned into 4 sub-blocks, the level increases, and the process is repeated. When the maximum partition level is reached, the best transformation found is stored, even if the pixel by pixel match error for the block exceeds 8. The quality settings of the JPEG encoding for *low*, *medium*, and *high* level of detail were 20, 40 and 60 respectively.

Taking into account that there are 3 images available, 2 compression methods and 3 levels of detail per method, a total of 18 features are generated per image.

Table 1. Fractal image compression parameters

	low	medium	high
Image size	256×256 pixels		
Minimum partition level	2	2	3
Maximum partition level	4	5	6
Maximum error per pixel	8	8	8

4 Experiments

This section details two experiments related to aesthetics, (i) a classification one using two different approaches (ANNs and SVMs) and (ii) a sorting one. The dataset used in the first task is explained next.

4.1 Dataset

The features presented have been tested on a collection of images previously used for aesthetic classification tasks [4,11]. It is a large and diverse set of ranked photographs for training and testing available via `http://ritendra.weebly.com/aesthetics-datasets.html`. This address also provides more recent datasets, but we are not aware of any published results using them. All of these images were taken from the photography portal "photo.net". This website is an information exchange site for photography with more than 400,000 registered users. It comprises a photo gallery with millions of images taken by thousands of photographers. They can comment on the quality of the pictures by evaluating their aesthetic value and originality, assigning them a score between 1 and 7. The dataset included color and grayscale images. Additionally, some of the images have frames. None of these images was eliminated or processed. Because of the subjective nature of this problem, both classes were determined by the average user ratings.

This dataset includes 3581 images. All the images were evaluated by at least two persons. Unfortunately, the statistical information from each image, namely number of votes, value of each vote, etc., is not available. Like in the previous approaches, they considered two image categories: the most valued images (average aesthetic value ≥ 5.8, a total of 832 images) and the least valued ones (≤ 4.2, a total of 760 images), according to the ratings given by the users of the portal. Images with intermediate scores were discarded. Datta's justification for making this division is that photographs with an intermediate value "are not likely to have any distinguishing feature, and may merely be representing the noise in the whole peer-rating process" [4]. However, when we carried out our experiment, some of the images used by Datta were not longer available at "photo.net", which means that our image set is slightly smaller. We were able to download 656 images with a rating of 4.2 or less, and 757 images with a rating of 5.8 or more. Out of the available images, about 7.4% are in grayscale.

4.2 Aesthetic Classification

The difference existing between the dataset of Datta et al. and the proposed one as regards the number of images used makes it impossible to compare the results. Having the input data of his experiment, as well as the input parameters, we have reproduced his experiment using only those images that we were able to retrieve. They perform classification using the standard RBF Kernel ($\gamma = 3.7$, *cost* =1.0) using the LibSVM package [3] and a 5-fold cross-validation (5-CV). Their success rate using this configuration was 70.12%. On our behalf, with their input data and the images available, 71.44% of images are classified correctly. The difference between both results shows that the task performed in this paper is less complicated than the original one. We will compare our results with the latter from now on.

We have used two different approaches in order to compare the functioning of the metrics proposed. One of them is based on Support Vector Machines (SVMs), while the other one is based on Artificial Neural Networks (ANNs). In the case of SVMs, we have decided to use the standard Linear Kernel configuration using the LibSVM package [19][3]. The success rate achieved in that case was 72.43%.

The other classifier is composed of a feed-forward ANN with one hidden layer. For training purposes, we resorted to SNNS [21] and standard back-propagation. The values that result from the feature extractor are normalized between 0 and 1. The results presented in this paper concern ANNs with one input unit per feature, 12 units in the hidden layer, and 2 units in the output layer (one for each category). A training pattern specifying an output of $(0, 1)$ indicates that the corresponding image belongs to the "low quality" set. Likewise, a training pattern with an output of $(1, 0)$ indicates that the corresponding image belongs to the "high quality" set. For each experiment we perform 50 independent repetitions of the training stage so as to obtain statistically significant results. For each of these repetitions we randomly create training, test, and validation sets with respectively 80%, 5%, and 15% of the patterns. The training of the ANNs is halted at 400 training cycles, or an RMSE in both the training and test sets lower than 0.01 is reached. Some other parameters used are shown in table 2. The results obtained with ANNs are very similar to those of SVMs, with a validation success rate of 71.16%.

Table 2. Parameters relative to the ANNs

Parameter	Setting
Init. of weights	random, $[-0.1, 0.1]$
Learning rate	0.15
Shuffle weights	yes
Class distribution	one-to-one
Max. tolerated error	0.3

4.3 Image Ranking

We will try to show the aptness of our metrics visually by showing the sorting capacity of the images obtained from a web search application and previously used by Ke et al. [10]. They used Google and Flickr to search for six image sets, labeled "apple", "bmw", "cow", "rose", "Statue of Liberty", and "violin". The retrieved images were then ranked by their quality assessment algorithm with a success rate of 72% obtained with a dataset of 12,000 images coming from the photography portal "DPChallenge.com".

The advantage of using a neural network lies in achieving two continuous outputs with values that can be used for another purpose, for instance, as fitness function determining the aesthetic quality of a particular image. In our case, we will use both neural network outputs in order to create the formula 2 which will be used as sorting criterion, having been used by [13]:

$$\frac{(O_1 - O_2) + 1}{2} \tag{2}$$

In this case, O_1 and O_2 will correspond to the ANN outputs. In case the first one has a high value, the ranking value obtained will be close to 1, which indicates, in our case, a high aesthetic quality. However, in case the value of the second output is higher, then the ranking value will be close to 0, indicating a low aesthetic quality. When $O_1 = O_2$ the ranking value will be 0.5.

Following the approach of Ke et al. [10], in Figure 1 displays ends of the sorting, that is, the three best and the three worst. It is also important to observe what happens in the intermediate area of the ranking. In Figure 2 we present the entire list of images from the gallery retrieved by the search word "rose" sorted accordingly to formula 2. The full sorted lists of each of the 6 image sets are available on the Internet at `http://193.147.35.124/papers/evomusart2011`.

Taking into account the network outputs and the formula proposed, the values given to each image should be distributed in a space with range [0,1]. Due to the training model proposed for the ANN, the interval [0, 0.3] equals 0 and the interval [0.7, 1] equals 1. Thanks to that, the network output models can have a more linear approach, thus allowing the exploration of the ends, as done by [14]. In that particular case, the end values seen in Figure 1 are located within the range [0.85, 0.25]

In the subjective perspective of authors, the sorting achieved is far from perfect but quite successful from the point of view of aesthetics, particularly in what concerns the "best" and "worst" images of each set, albeit some isolated exceptions. One of these exceptions is "Statue11", which we consider as one of the best images of the subset.

By analyzing the sorted lists produced by the proposed approach one can try understand how the rankings are being determined. The results indicate that the best valued images tend to be those where the difference between the figure and the background is more evident, as well as those that have high contrast. It seems that two of the most determining elements are: the simplicity of the background (either due to flat elements or due to a low depth of field leading to an unfocused background); the existence of a significant difference between the background and the figure in the foreground. The image contrast can be also a decisive element, together with the existence of pure white and deep black, and a well-balanced distribution of both. For instance, image "Cow33" in Figure 1 has deviation similar to the deviations of the best valued high-contrast images, however, unlike them, it is underexposed, which causes a lack of information in the highlights and a trimming in the shadows, making it harder to differentiate between the background and the figure.

The rankings produced cannot be fully explained by these factors alone and the exact sorting method of the system is far from being understood.

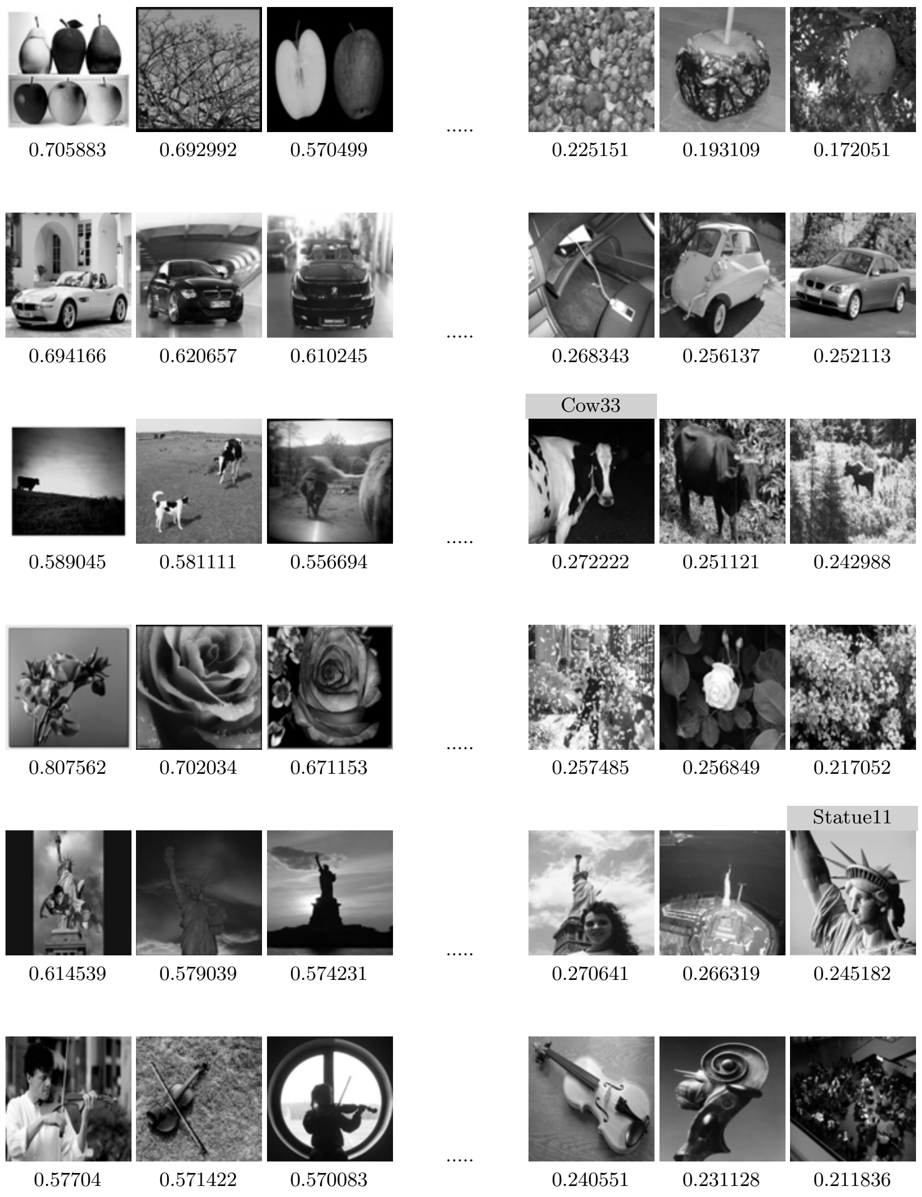


Fig. 1. End images of each gallery with its associated aesthetic value. Each set is shown in a row, with the three "best" images on the left and the three "worst" on the right.

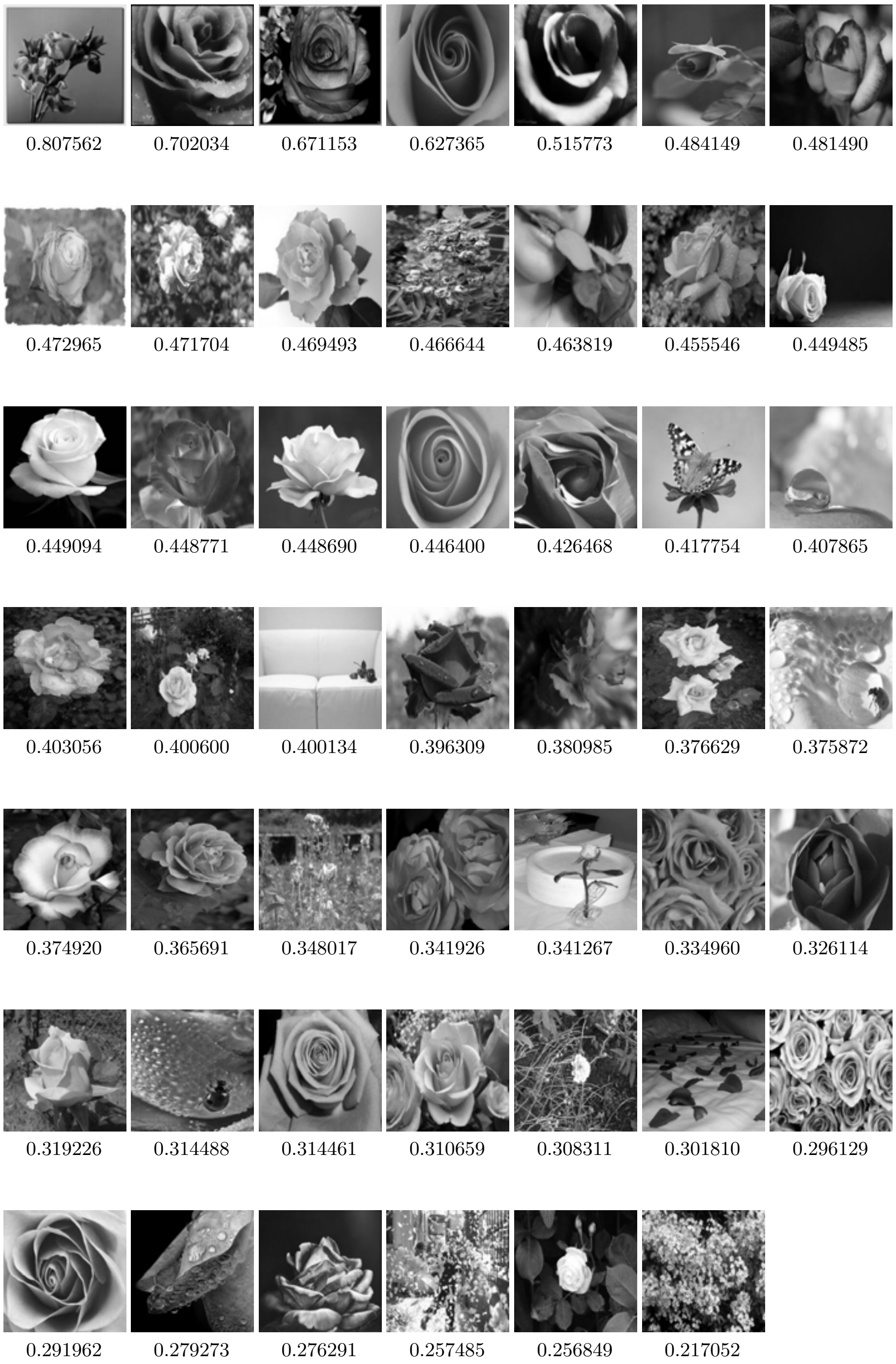


Fig. 2. Whole sorting list of the image gallery "rose"

Therefore, we can state that among the worst classified images most of them have brightness levels tending towards a concentration at the medium values of the image, together with over and underexposed ones.

5 Conclusions and Future Work

It has been shown how a set of 18 metrics based on two widespread compression methods can be used for image classification and sorting tasks. An experiment of aesthetic classification of images was carried out achieving similar results to other ad-hoc metrics specifically developed for that purpose, using two different approaches: one of them based on SVMs and the other one based on ANNs. A sorting function based on the output of the ANN used in the classification experiment was proposed and its functioning when sorting particular image sets based on aesthetic criteria presented and discussed.

In the future, the research will be expanded to cover other metrics related to complexity in both tasks. The purpose is using a large set of metrics to develop a fitness function within our own evolutionary engine.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments, suggestions and criticisms. This research is partially funded by: the Portuguese Foundation for Science and Technology, research project PTDC/EIAEIA/115667/2009; the Spanish Ministry for Science and Technology, research project TIN200806562/TIN; Xunta de Galicia, research project XUGAPGIDIT10TIC105008PR.

References

1. Arnheim, R.: Art and Visual Perception, a psychology of the creative eye. Faber and Faber, London (1956)
2. Birkhoff, G.D.: Aesthetic Measure. Harvard University Press, Cambridge (1932)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
4. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Studying aesthetics in photographic images using a computational approach. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3953, pp. 288–301. Springer, Heidelberg (2006)
5. Eysenck, H.J.: The empirical determination of an aesthetic formula. Psychological Review 48, 83–92 (1941)
6. Eysenck, H.J.: The experimental study of the 'Good Gestalt' - A new approach. Psychological Review 49, 344–363 (1942)
7. Fisher, Y. (ed.): Fractal Image Compression: Theory and Application. Springer, London (1995)
8. Graves, M.: Design Judgment Test. The Psychological Corporation, New York (1948)

9. Greenfield, G., Machado, P.: Simulating artist and critic dynamics - an agent-based application of an evolutionary art system. In: Dourado, A., Rosa, A.C., Madani, K. (eds.) IJCCI, pp. 190–197. INSTICC Press (2009)
10. Ke, Y., Tang, X., Jing, F.: The Design of High-Level Features for Photo Quality Assessment. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 419–426 (2006)
11. Luo, Y., Tang, X.: Photo and video quality evaluation: Focusing on the subject. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 386–399. Springer, Heidelberg (2008)
12. Machado, P., Cardoso, A.: Computing aesthetics. In: de Oliveira, F.M. (ed.) SBIA 1998. LNCS (LNAI), vol. 1515, pp. 219–229. Springer, Heidelberg (1998)
13. Machado, P., Romero, J., Manaris, B.: Experiments in Computational Aesthetics. In: The Art of Artificicial Evolution. Springer, Heidelberg (2007)
14. Machado, P., Romero, J., Manaris, B.: Experiments in computational aesthetics: An iterative approach to stylistic change in evolutionary art. In: Romero, J., Machado, P. (eds.) The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music, pp. 381–415. Springer, Heidelberg (2007)
15. Machado, P., Romero, J., Santos, A., Cardoso, A., Manaris, B.: Adaptive critics for evolutionary artists. 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. 435–444. Springer, Heidelberg (2004)
16. Meier, N.C.: Art in human affairs. McGraw-Hill, New York (1942)
17. Moles, A.: Theorie de l'information et perception esthetique, Denoel (1958)
18. Tong, H., Li, M., Zhang, H., He, J., Zhang, C.: Classification of Digital Photos Taken by Photographers or Home Users. In: Aizawa, K., Nakamura, Y., Satoh, S. (eds.) PCM (1). LNCS, vol. 3332, pp. 198–205. Springer, Heidelberg (2004)
19. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with java implementations. SIGMOD Rec. 31(1), 76–77 (2002)
20. Wong, L., Low, K.: Saliency-enhanced image aesthetics class prediction. In: ICIP 2009, pp. 997–1000. IEEE, Los Alamitos (2009)
21. Zell, A., Mamier, G., Vogt, M., Mache, N., Hübner, R., Döring, S., Herrmann, K.U., Soyez, T., Schmalzl, M., Sommer, T., et al.: SNNS: Stuttgart Neural Network Simulator User Manual, version 4.2. Tech. Rep. 3/92, University of Stuttgart, Stuttgart (2003)

iSoundScape: Adaptive Walk on a Fitness Soundscape

Reiji Suzuki[1,2], Souichiro Yamaguchi[1], Martin L. Cody[2], Charles E. Taylor[2], and Takaya Arita[1]

[1] Graduate School of Information Science / SIS, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
[2] Department of Ecology and Evolutionary Biology
University of California
Los Angeles, CA, 90095, USA

Abstract. Adaptive walk on a fitness soundscape [7] is a new kind of interactive evolutionary computation for musical works. This system provides a virtual two-dimensional grid called a "soundscape" in which each point corresponds to a genotype that generates a sound environment. By using the human abilities of localization and selective listening, the user can "walk" toward genotypes that generate more favorable sounds. This corresponds to a hill-climbing process on the "fitness soundscape." This environment can be realized by multiple speakers or a headphone creating "surround sound." In this work we describe two new applications of adaptive walk. The first is developed for creating spatially grounded musical pieces as an interactive art based on fitness soundscapes. The second provides a new way to explore the ecology and evolution of bird songs, from scientific and educational viewpoints, by exploring the ecological space of "nature's music", produced by populations of virtual songbirds.

Keywords: interactive evolutionary computation, musical composition, fitness landscape, surround sound, birdsongs, artificial life.

1 Introduction

Interactive evolutionary computation (IEC) has been used for optimizing a variety of artifacts which cannot be evaluated mechanically or automatically [8]. Based on subjective evaluations by a human, one's favorite artifacts in the population are selected as "parents" for new artifacts in the next generation. By iterating this process, one can obtain better artifacts without constructing them directly.

IEC has found use in a variety of artistic fields, including visual displays [6], musical compositions [9] and sound design [3]. While promising, IEC for musical composition has shortcomings because it is difficult to evaluate candidate sounds when a large number of them are played at once. Consequently, the users typically had to listen to each piece separately and evaluated them, one by one. This sequential evaluation of individuals has two costs: an increase in the total

© Springer-Verlag Berlin Heidelberg 2011

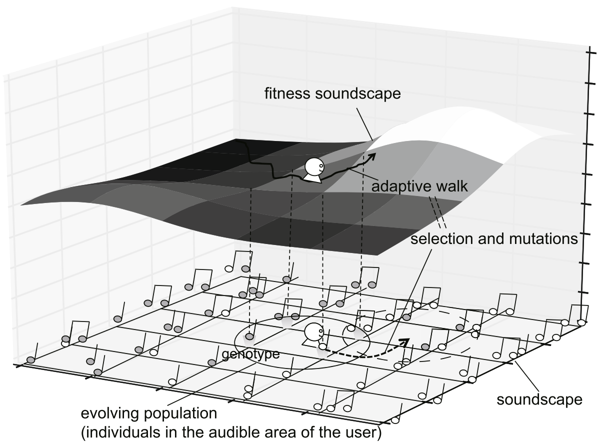


Fig. 1. A basic concept of adaptive walk on a fitness soundscape and its relationship with evolutionary computation

evaluation time (temporal cost), and the need for remembering the features of individuals in order to compare them (cognitive cost). Thus, it is necessary to limit the population of candidate pieces in each generation to a small size in order to decrease these costs, leading to inefficient selection and introducing substantial random genetic drift [1].

Recently, several studies proposed and developed surround-sound-based browsing or exploration of musical collections or sounds [5,4]. Also, to increase efficiency of IEC for sounds, Suzuki and Arita [7] proposed a new kind of IEC for musical works which was inspired by a biological metaphor, adaptive walk on fitness landscapes. A fitness landscape is used to visualize and intuitively understand the evolutionary dynamics of a population [10]. Recognizing that similar sounds could be placed nearby one another in a virtual landscape, they constructed a system that combined human abilities for localization and selective listening of sounds with a hill-climbing process on fitness soundscapes. This system enables a user to explore her favorite musical works by moving through a virtual landscape of sounds. Fig. 1 shows a conceptual image of this system. It may be summarized as follows:

– We assume a set of genotypes that can describe all possible musical pieces to be explored.
– We also assume a two-dimensional grid, and map every genotype to a unique grid point so that there is a correlation between the distance and similarity among genotypes on the grid. That is to say, similar genotypes are located nearby, while less similar ones are more distant in this grid.
– Each genotype can play its own musical pieces at its corresponding location in the grid. The resulting two-dimensional acoustic space is called “soundscape”, as shown in Fig. 1 (bottom).

– A user of this system, whom we will call the listener, has a location in the soundscape, and can hear the sounds of neighboring genotypes at the same time, if desired. The sounds come from different directions that correspond to their locations on the soundscape. This virtual environment can be realized by a multiple speaker system creating "surround sound."
– Humans have a sophisticated ability to localize the direction of sounds, and can focus their attention in one direction or another. This is sometimes called the *cocktail party effect* [2]. By using their ability to localize, the listener can evaluate neighboring sounds at the same time. Their evaluation of goodness of the sounds gives each direction an analog to a gradient of fitness, thereby defining a third dimension – which may be thought of as a fitness surface on the soundscape – a "fitness soundscape." The actual shape of the fitness soundscape will be different among listeners depending on their subjective impression, and can also change dynamically even for a single user.
– The listener is able to change her location – "walk" – along her own fitness soundscape by repeatedly moving toward the direction with an increasing fitness on the surface – i.e. in the direction from which more favored sounds are coming. This corresponds to the evolutionary process of the population in standard IECs. In other words, adaptive evolution of the population can be represented by a walk along the soundscape, determined by a hill-climbing process on the fitness soundscape.

This system can be regarded as a kind of evolutionary computation in the following sense: An evolving population in evolutionary computation corresponds to the set of genotypes on the soundscape whose sounds can reach the user. The movement of the user toward her favorable sounds corresponds to selection and mutation operations because less favorable genotypes disappear and new, slightly, different genotypes appear in the new population due to the shift of the audible area of the user. Although a crossover or recombination operation is not incorporated into the conceptual model, changing the scale and shape of the soundscape implemented in the prototype can contribute to maintaining the genetic diversity of the population (explained later).

Suzuki and Arita [7] constructed a prototype of the system using a personal computer to control a multi-channel home theater system with 7 speakers. They confirmed that listeners were able to search for their subjectively more favorable pieces by using this system. Their experience led to a proposal for improving evolutionary search in that system. Finally, they noted that a searching process on the soundscape was itself a new experience for the listener, suggesting that the system might be implemented as an art installation.

In this paper, we report on two variant applications of this concept. The first is developed for creating more spatially grounded musical pieces as a kind of interactive art, exploring better use of the surround sound environments with an iPhone or iPad. The second variant is to provide a new way to experiment with the ecology and evolution of birdsongs from both scientific and educational viewpoints — in a sense to "nature's music" by songbirds.

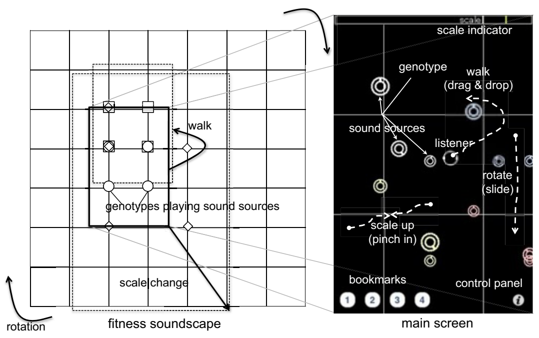


Fig. 2. Basic operation of the iSoundScape App (left) and a snapshot (right) of the main screen, with labels described in the text

iPhones and iPads provide a variety of sensors and interfaces, and have a rich software development library for sound works, such as Core Audio and OpenAL. The devices are now becoming standard platforms for applications of interactive arts based on sounds. For example, Brian Eno, a pioneer of ambient music, recently released a kind of interactive music box called "Bloom", in which a user can create drones playing periodically by touching a screen. Bloom and several variants are available at http://www.generativemusic.com/. In the applications described here, we use only a stereo (two-speaker) audio environment, rather than the richer environment provided by the 7-speakers in earlier studies. Something is lost, certainly, but we are finding that the intuitive operation and quick responses of a multi-touch interface may at the same time bring new benefits for exploring fitness soundscapes. The availability of world-wide channels to distribute applications from AppStore and to obtain feed-back from users are also benefits from this platform.

2 iSoundScape

We constructed an application for iPhone termed iSoundScape based on an adaptive walk on the fitness soundscape illustrated in Fig. 2. It was developed using Objective-C++ with XCode, and enlists the OpenAL API for creating surrounded sounds. It is currently available from the Apple App store at no cost.

Fig. 2 (left) shows a global image of a soundscape represented as a two-dimensional and toroidal grid space in this system. Fig. 2 (right) is a snapshot of a main screen of the implementation of an iPhone. It shows a part of the whole soundscape from a top view. An open circle in the center corresponds to the listener and generates her sound environment there. There are also several double circles that represent the sound sources mapped from genotypes on nearby grid points. The listener is assumed to face upward, and can hear the sounds emanating from the corresponding sources, in a manner described below.

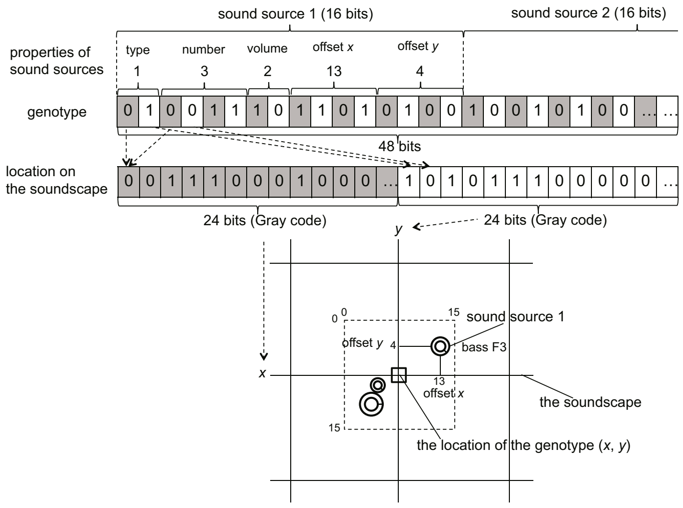


Fig. 3. The relationship between a genotype and its location on the soundscape, and one mapping from a genotype to a musical piece

2.1 Genotypes for Creating a Soundscape

Spatially grounded representations of musical pieces are utilized in this work, improving the use of the acoustic space of the soundscape compared to the simple scores often used in previous research [9,7].

Fig. 3 shows the relationship between a genotype and its location on the soundscape, as well as one mapping from a genotype to its musical piece. Each genotype is composed of a bit string with a length of 48, which contains enough information to code for more than 67 billion possible genotypes. We can imagine each 48-bit string as a point in a $2^{24} \times 2^{24}$ soundscape grid. The information of a genotype is used to determine its location on the soundscape, and also used to determine the properties of its corresponding musical piece as described below.

In order to determine the location on the soundscape on the $2^{24} \times 2^{24}$ grid, the bit string of a genotype is split into the two substrings composed of odd or even bits in the original genotype respectively, as shown in Fig. 3. Each substring is mapped to an integer value by using a Gray code (Reflected Binary Code), and the set of integer values is defined as the x and y location of the genotype on the soundscape. A Gray code maps each number in the sequence of integers $\{0, 1, \cdots, 2^{24}-1\}$ as a bit string with the length 24 so that consecutive numbers have binary representations that each differ by one bit. In this way, genotypes are closely correlated to their locations on the soundscape.

Each genotype generates and plays a musical piece distinctive to its location on the soundscape, composed of sounds from the three closest sound sources played on an infinite loop. In this prototype, the four nearest genotypes displayed on the screen generate the sound sources. In order to determine the property of these sound sources, the bit string of the genotype is also split into the three

Table 1. The sound clips

ID	type	unique number
0–15	piano (0)	C3–C5 (0–14) and no sound (15)
16–31	bass (1)	C3–C5 (16–30) and no sound (31)
32–47	synthesizer (2)	C3–C5 (32–46) and no sound (47)
48–63	drums / sound effects (3)	drums (48–55), singing birds (56, 57), sound of water (58, 62), call of goat (59), laughter (60), clock (61) and ambient sound at lobby (63)

substrings with the length 16, as shown in Fig. 3. Each substring determines a type of sound clip to play (4 types), a unique number in the type (16 different clips), a volume (4 levels), and a relative position from the grid point of the genotype (16 × 16 locations) on the soundscape, as shown in Fig. 3. The type of sound and the unique number in the type specifies one of 64 different sound clips (in wave format) as listed in Table 1. Piano, bass and synthesizer are each characterized as a short drone with a unique pitch. We also prepared some beats on drums and sound effects from various natural or social environments. Each sound source plays its sound clip with the specified volume repeatedly. Either even or odd bits are used from a genotype to determine the appropriate x or y location by relating each x and y value to the overall properties of the sound sources. Each sound source is represented as a double circle with a tab that indicates the ID of the sound clip. The size of the circle represents the volume of the sound source.

2.2 Operations for Adaptive Walk on a Fitness Soundscape

There are several basic operations to walk on the soundscape as follows:

Adaptive walk. After evaluating the musical pieces generated by the nearest four genotypes, a listener can change its location on the soundscape by dragging its icon. The directions of sounds coming from sound sources change dynamically according to the relative position of the listener during the movement. If the icon is moved outside of the central area of the grid, the soundscape scrolls by one unit of the grid as shown in Fig. 2 (left). The sound sources of the next four neighboring genotypes are then displayed and begin to play.

Rotation. By swiping a finger outside of the center area in a clockwise or counterclockwise direction, a listener can rotate the whole soundscape by 90 degrees (Fig. 2 left). By changing the orientation of the surrounding sound sources in this way, a listener can utilize the right / left stereo environment of the iPhone for localizing sounds that were previously in front of / behind the listener.

Scale change of the soundscape. By pinching in or out any place on the screen, a listener can decrease or increase the scale of the soundscape. It changes the Hamming distance between the nearest neighboring genotypes as shown in Fig. 2 (left) by skipping the closest genotypes and selecting distant genotypes. The decrease in the scale ratio enables the listener to evaluate more different individuals at the same time, and jump to a more distant place quickly. Conversely, increasing the scale ratio allows the listener to refine the existing musical pieces by evaluating more similar genotypes.

Shape change of the soundscape. A user can change the shape of the soundscape by modifying the genotype-phenotype mapping shown in Fig. 3. Every time the user shakes the iPhone or iPad, the position of each bit in the genotype that corresponds to each property of the sound sources is right-shifted cyclically by two bits. Then, the user jumps to the location on the soundscape based on the new genotype-phenotype mapping so that the sound sources of the genotype in the user's front left position are kept unchanged. This enables the user to explore a wide variety of mutants of the current population.

Bookmark of a location. If a listener touches one of the four buttons on the bottom left in the screen, they can save the current position, scale and direction of the soundscape as a kind of bookmark. One previously bookmarked state can be loaded as the current state of the user.

Finally, a listener can change some optional settings to facilitate exploring processes by touching the button on the bottom right.

2.3 Basic Evaluations

iSoundScape has been freely available from the AppStore[1], an online store for downloading applications for iPhone. Approximately 1,100 people around the world have downloaded it since May 2010. Users who have commented on iSoundScape have made several suggestions. The most important of these relate to orientation on the fitness landscape. The first suggestion is that stereo headphones or external speakers were effective for localizing sounds from right and left directions, although it was not always easy to localize sounds coming from in front of or behind the listener. Second, it was helpful to actively move back and forth on the soundscape for localizing the sounds. With these methods, listeners were able to effectively evaluate each musical piece and search for favorable musical pieces without difficulty.

The general consensus of the user comments was that the spatially grounded representation of musical pieces worked well on the soundscape and created a new kind of listening experience invoking a feeling that one is surrounded by different elements of a musical piece. A novel aspect of this experience is that a set of sound sources from neighboring genotypes is interpreted as a kind of musical work of itself. The wide variety of sound types including sound effects produced through the exploration process itself provides an entertaining and engaging experience because small changes in the sound types and locations can change the aesthetic feeling of the musical piece in unexpected ways. Also, the

[1] http://itunes.apple.com/us/app/isoundscape/id373769396?mt=8

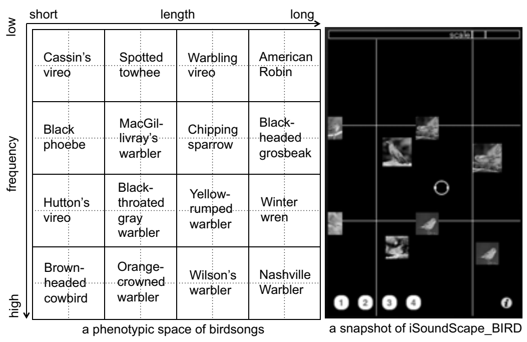


Fig. 4. A phenotypic space of birdsongs and a snapshot of iSoundScape_BIRD

scale or shape change of the soundscape enables a user to jump from the current location to distant locations at one time. This can be a helpful method to explore the whole soundscape quickly.

3 iSoundScape_BIRD

To take the system a step further, we developed a variant of iSoundScape termed iSoundScape_BIRD that features various birdsongs from California. There are several reasons to feature birdsongs. First, birdsong is one of the best examples of "nature's music", and may appeal to some people on this level. Second, there is the potential for practical applications of this system in exploring the ecology and evolution of birdsongs from a new perspective. For example, the parallel and simultaneous listening of songs from various directions is analogous to some aspects of natural soundscapes. Lastly, this system could provide a new way to teach concepts of adaptive evolution in an educational environment. Here, we enumerate features modified from the original iSoundScape.

We prepared 64 sound clips of birdsongs from the sound tracks recorded in Amador County, in Northern California. There are 16 species as shown in Fig. 4 (left), and 4 different song clips for each species. The duration of songs varied from about 0.5 to 7.0 seconds. The song properties varied significantly from this single location.

We have constructed two different soundscapes from these song clips. Each reflects a phenotypic or ecological property of birdsongs. In addition to providing an interesting way to explore birdsongs, the first soundscape was developed to provide a way to understand how properties of birdsongs could vary among species in a soundscape. We assumed a two-dimensional phenotypic space of birdsongs as shown in Fig. 4 (left). The axes in this space reflect roughly two basic properties of birdsongs: the length of songs and the intermediate frequency of songs. Instead of using genotypes, we directly mapped the all song clips to a 8×8 two-dimensional soundscape so that squared clusters of four song clips of each species are arranged according to the phenotypic space in Fig. 4 (left). Thus, each genotype is composed of a set of two integer values, each corresponding to the x or y location within the phenotypic space.

In a second form, we replaced the original sound clips in the iSoundScape with the clips of birdsongs. To do so, we assumed that the first 4 bits in the substring for each sound source in a genotype represent the species, reflecting the topology of the phenotypic space in Fig. 4, and the next 2 bits represent a unique number of song clips in the species. We also changed the number of sounds for each genotype from three to two. Thus, the soundscape comprises a $2^{16} \times 2^{16}$ grid. In this case, each genotype is composed of a 32 length bit string representing an ecological situation involving two birds, and is mapped to both individual birds singing different songs at different locations. Thus, a listener is able to explore the soundscape and virtual ecology of multiple individuals of songbirds.

In both cases, we inserted a random interval between the start times of songs from each bird to approximate the variation found in many natural soundscapes.

Finally, we used a small picture to represent the species of each individual[2]. This allows the listener to recognize the distribution of birds on the soundscape.

3.1 Preliminary Evaluations

Fig. 4 (right) shows a snapshot of iSoundScape_BIRD. In our preliminary evaluations with the phenotypic soundscape, we could recognize changes in the length and frequency of songs gradually through exploration of the soundscape, and understand how these species have different types of songs. In addition, it was helpful to recognize the difference between the similar but slightly different songs of different species because they are closely located on the soundscape and the listener can hear their songs from different directions at the same time to compare their properties between them. In the case of the ecological soundscape, the listener could feel the acoustic environment as more ecologically realistic because more birds were singing at different locations.

We believe both can provide a new way to understand ecology and evolution of birdsongs from scientific and educational viewpoints in addition to an artistic point of view. We are planning to release this variant at AppStore.

4 Conclusion

We proposed two variants of interactive evolutionary computation for musical works based on adaptive walk on a fitness soundscape. The first variant was developed to explore a new application of IECs for creating spatially grounded musical pieces as a new kind of interactive art, based on the concept of soundscape. Listeners were able to search for their favorite musical pieces by moving around

[2] These pictures are provided by Neil Losin and Greg Gillson from their collections of bird pictures (http://www.neillosin.com/, http://www.pbase.com/gregbirder/)

the soundscape actively, even within the limitations of two speakers for localization. The second variant was developed to explore soundscapes of songbirds to explore new ways to experiment with the ecology and evolution of birdsongs. It appears that this system may find application to better understand properties and ecology of birdsongs by creating phenotypic and ecological soundscapes.

Future work includes more detailed evaluations of these applications, a use of other sensors in the device such as GPS or an acceleration meter in order to make the system more interactive, and an addition of some functions that enable us to learn more about songbirds in the system.

Acknowledgements

The authors thank Zac Harlow, who gave fruitful comments and advice for development of iSoundScape_BIRD, and Neil Losin and Greg Gillson, who allowed us to use the beautiful pictures of songbirds from their collections. This work was supported in part by the Excellent Young Researcher Overseas Visit Program of the Japan Society for the Promotion of Science.

References

1. Biles, J.A.: GenJam: A Genetic Algorithms for Generating Jazz Solos. In: Proceedings of the 1994 International Computer Music Conference (1994)
2. Cherry, E.C.: Some Experiments on the Recognition of Speech with One and with Two Ears. Journal of the Acoustical Society of America 25, 975–979 (1953)
3. Dahlstedt, P.: Creating and Exploring Huge Parameter Spaces: Interactive Evolution as a Tool for Sound Generation. In: Proceedings of the International Computer Music Conference 2001, pp. 235–242 (2001)
4. Knees, P., Schedl, M., Pohle, T., Widmer, G.: Exploring Music Collections in Virtual Landscapes. IEEE Multimedia 14(3), 46–54 (2007)
5. Rocchesso, D., Bresin, R., Fernström, M.: Sounding Objects. IEEE Multimedia 10(2), 42–52 (2003)
6. Sims, K.: Interactive Evolution of Dynamical Systems. In: Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pp. 171–178 (1992)
7. Suzuki, R., Arita, T.: Adaptive Walk on Fitness Soundscape. In: Proceedings of the Tenth European Conferences on Artificial Life (ECAL 2009). LNCS, LNAI (2009) (in press)
8. Takagi, H.: Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. Proceedings of the IEEE 89(9), 1275–1296 (2001)
9. Unemi, T.: SBEAT3: A Tool for Multi-part Music Composition by Simulated Breeding. In: Proceedings of the Eighth International Conference on Artificial Life, pp. 410–413 (2002)
10. Wright, S.: The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in Evolution. In: Proceedings of the Sixth International Congress on Genetics, pp. 355–366 (1932)

The *T. albipennis* Sand Painting Artists

Paulo Urbano

LabMag—Faculdade Ciências da Universidade de Lisboa
pub@di.fc.ul.pt

Abstract. In recent years, we have seen some artificial artistic work that has drawn inspiration from swarm societies, in particular ant societies. Ant paintings are abstract images corresponding to visualizations of the paths made by a group of virtual ants on a bi-dimensional space. The research on ant paintings has been focused around a stigmergic mechanism of interaction: the deposition of pheromones, largely used by ants. In an effort to further on the research on ant inspired artificial art, we introduce the *T. albipennis* sand painting artists, which draw direct inspiration from the ant species *Temnothorax albipennis* (formerly *tuberointerruptus*). These ants build simple circular walls, composed of grains of sand or fragments of stones, at a given distance from the central cluster of adult ants and brood. The brood and ants cluster function as a template, which combined with self-organization are responsible for the particular wall pattern formation. The *T. albipennis* artists are artificial two-dimensional builders, starting from unorganized placement of virtual sand grains, they rearrange them, creating some interesting patterns composed of scattered pointillistic and imperfect circles, a colored moon-like landscape full of craters.

Keywords: Ant Paintigs, Swarm Art.

1 Introduction

Social insects perform a large variety of extraordinarily complex collective tasks, being nest building the most spectacular [4]. Social insects colonies are a decentralized systems composed of autonomous and cooperative individuals that are cognitively very simple, exhibiting simple probabilistic stimulus-response behavior and which have access to limited local information. Ants have been, since the beginning of the 21th century [1, 7, 15, 8, 9], a source of inspiration for media arts, where artists seek to create a collection of autonomous artificial entities, which are able to interact directly or indirectly, in stochastic ways and produce emergent and unexpected patterns with some aesthetical value.

Ant paintings, a term first used by Aupetit et al. [1], are abstract images made on background neutral color, corresponding to visualizations of the paths made by a group of virtual ants that wander around on a toroidal virtual canvas, leaving colored traces. They used small virtual ant colonies (typically between 4 and 6) where the individuals do not communicate directly with each other—they use a stigmergic mechanism where color plays the main role. The luminance color value functions as a

© Springer-Verlag Berlin Heidelberg 2011

virtual pheromone, which controls the movement of the ants and consequently their painting activity. The virtual scent has no evaporation and presentes limited diffusion properties. Each ant, as it moves, lays down one color while searching for the color deposited by other ants. Ant behavior is represented by a genome that determines what color ants should deposit, what color ants should seek, and their movement characteristics. Ant paintings were evolved using an interactive genetic algorithm. Greenfield [7] extended the previously referred work, using larger ant populations (8-12), introduced perception of tristimulus color values instead of luminance, and has designed a non-interactive genetic algorithm for evolving ant paintings. The fitness function he has designed measures the exploration and exploitation capabilities of the ants. Urbano [15] introduced a new model: color has no scent properties and it is the reticular canvas that produces an invisible virtual scent, responsible for attracting individuals, influencing their movement and consequent traces. The population of micro-painters can be much more numerous (thousands of virtual ants). The individual ant traces are soon very difficult to follow and what is visualized is the exploration global pattern of the colony of ant painters. Each cell of the environment diffuses its chemical, which is subject to evaporation, but to be a chemical producer a cell must satisfy certain conditions. He has considered two variations on the model: either a non-painted cell produces scent (Colombines style) or a painted cell is the chemical producer (Anti-Colombines style). The visual characteristics of these ant paintings are influenced both by the number of ants and their initial placement. Greenfield [8, 9] considered simulated groups of ants whose movements and behaviors are influenced by both an external environmentally generated pheromone and an internal ant generated pheromone making a blend of the model in [1, 7] and the model in [15].

Outside of the scope of ant painting but of related interest, Urbano [16] has investigated consensual decision making in swarm paintings, where simple artificial artists use a distributed coordination mechanism to decide the values of decisive attributes for painting activity, like color, orientation or speed. Jacob and Hushlack were inspired by the boids model of Reynolds [13] to make swarm Art [14, 3]. Moura and Ramos [11] used autonomous robots to make real paintings.

Trying to further on the research on ant inspired artificial art, we introduce the *T. albipennis* colony of artificial artists, which produce virtual sand paintings. They draw direct inspiration from the ant species *Temnothorax albipennis* (formerly *tuberointerruptus*). These ants build simple perimeter walls, composed of grains of sand or fragments of stones, at a given distance from the central cluster of ants and brood. The brood and ants cluster function as a template, which combined with self-organization are responsible for the particular wall pattern formation.

The wall builder ants have also inspired research on robotics, with no relation with art, but we think it deserves to be mentioned here. Melhuish et al. [10] developed a group of minimalist robots able to build a roughly linear wall composed of pucks, using as cues a white line on the floor and a bank of halogen lights. Parker, Zhang and Kube [12] created a group of blind bulldozing robots for debris cleanup. These robots equipped with force and collision sensors are able, by simply moving straight ahead and turning in random directions, to clean a space full of debris, simulated by pucks, which are pushed until the force opposing their movement exceeds a threshold.

The *T. albipennis* artists are artificial two-dimensional builders, which are able to create some interesting patterns composed of scattered pointillistic and imperfect circles. In this preliminary research we are going to use only the template effect, leaving the "grains attracting grains" effect for future developments. The visual characteristics of ant sand paintings are influenced by: the colony size, the number of existing virtual sand grains and their initial placement. This form of sand painting is a conservative one, the painting is made using only the initial available materials, which are recombined to form circles, the grains do not disappear and are not created either, during the painting process.

The paper is organized as follows: Section 2 describes wall building behavior of *Temnothorax albipennis* ant colonies and presents the two main mechanisms responsible for pattern formation: template and self-organization. In section 3 we present the *T. albipennis* artists. In section 4 we show some examples of *T. albipennis* art pieces and we conclude pointing out some future directions of research.

2 Wall Building in *TEMNOTHORAX albipennis*

Franks et al. [5] and Franks & Deneubourg [6] have observed that the colonies of ant species *Temnothorax albipennis* nest in narrow crevices of rocks that have been fractured by weathering. They build a dense circular wall (with one or more entrances) made of debris (fragments of rocks, grains of sand or particles of earth) around a central cluster of queen, brood and workers, which function has a physical or chemical template.

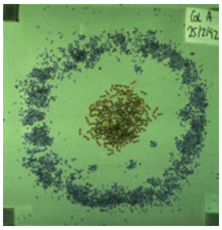


Fig. 1. Example of a nest built out of sand-like material by the *Temnothorax albipennis* in laboratory. We can see the circular wall surrounding a tight ant cluster and same entrances. Image courtesy of Nigel Franks and reproduced with his permission.

A template means that a blueprint of the nest 'already' exists in the environment. Although the exact nature of the template mechanism remains unknown, the nest size is adapted to the number of individuals belonging to the colony. In laboratory was seen [5] that when a colony migrates into a new nest slit, the brood is arranged at the center surrounded by the queen and the adults, which form a tight cluster. "Only when these formations of brood and adults are complete does a small minority of workers start building." [4] The template mechanism is not alone, combined with it there is a self-organizing mechanism. In fact, deposition behavior does not solely depend on the distance from the cluster of brood and workers. The local density of building materials plays also an important role during nest construction. "The probability of depositing a brick is highest when both the distance from the cluster is appropriate and the local density of bricks is large; it is lowest when the cluster is either too close or too far and when the local density of bricks is small. When the distance from the cluster does not lie within the appropriate range, deposition can nevertheless be observed if bricks are present; conversely, if the distance from the cluster is appropriate, deposition can take place even if the number of bricks is small." [2]

Camazine et al. [4] proposed a stochastic model where grain picking and dropping behavior is influenced both by the self-organizing amplifying effects (grains attract grains) and the influence of cluster of brood and workers template. Based on the model, they were able to simulate *Temnothorax albipennis* ant colonies' wall building behavior. As we have adopted their model for our *T. albipennis* artificial artists, the formulas for calculating the dropping and picking up probabilities, will be presented in the next section. They were slightly adapted to deal with multiple colonies and multiples walls.

3 The T. a*lbipennis* Artists

In our artistic appropriation of the behavior of *Temnothorax albipennis* we will only use the template rules. There will be no interaction between the artistic material (virtual sand grains, corresponding to fragments of stones, grains of sand or pieces of earth in the ants) and the artists—the virtual grain sands will not "attract" other grain sands and so positive feedback will play no role in the artistic process. In ants, the tight cluster of ants and brood functions as a physical or chemical template. In our model, we will not differentiate between artists, we will not have the equivalent to the queen or brood or nurse in ants—every individual is a wall builder. In order to adapt the *Temnothorax albipennis* wall formation behavior to the artistic realm we thought in using several colonies with different population dimensions and associating each one with a particular color. Each colony will build a wall of a certain color. The exact position of the template walls is decided randomly before the sand painting process begins. We have diverged from the real phenomenon in several aspects: we allow that an ant drops a piece of virtual sand grain inside the circle area corresponding to the tight cluster of queen, brood and nurses. As there is no cluster of ants in the center, the debris inside the nest center may be not cleared out. We are also not worried with collisions, neither between ants nor between ants and grains. This means that ants do not collide with each other, and can occupy the same cell. We do not need the formation of entrances for entering and leaving the nest. Ants can go through the wall.

3.1 The Black Sand Painting Reticular Canvas

We have considered a bi-dimensional reticular space with dynamic dimensions, but all the paintings here displayed were made in a canvas 401×401 which can be toroidal if we wish (some of the paintings here displayed were made in the toroidal and others on non-toroidal canvas). Each cell can take a grain of sand and the cell color corresponds directly to the grain color. A black cell is a cell with no grain and so there are no black colonies (no black walls either) because the background of the canvas is black. It would be very easy to enable different background colors.

3.2 Colonies of *T. albipennis* Artists

As in [4] we will not explicitly consider the physical or chemical influences of the worker plus-brood cluster, there will be a cluster center for each colony nest. Instead they have considered distance to the cluster center. Each *T. albipennis* artist belongs to a colony C_i and each colony C_i has a nest center c_i, which is a circle. Each nest center is a circle and is defined by a center: a point with coordinates (x_i y_i), and a radius ro_i. Note that colony nest centers are configured during sand-painting setup and remain unchanged). For a particular *T. albipennis* artist, the probability of picking or dropping a virtual grain sand in a particular zone of the artistic space, depends on the distance the zone is from the center of the ant respective cluster or colony center. A colony of *Temnothorax* ants, builds its nest in proportion to the existing population size—a bigger colony will build a bigger wall, further away from the colony center, considering a circular wall. Our artistic colonies will have different population sizes and thus they will "build walls" with different sizes, closer or further way from their nest centers.

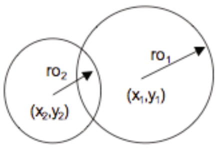


Fig. 2. Example of two colonies nest centers

In order to create polychromatic artistic pieces each colony is associated with a color although two different colonies might be associated with the same color. This means that an ant belonging to a green colony will try to make a green wall, made of green virtual grains of sand and two green colonies compete for the virtual green grains. Ants may destroy the wall of a similar colony in order to collect grains to build their own nest walls.

3.3 *T. albipennis* Artist Attributes, Perception Abilities and Behavior

The state of a *T. albipennis* sand artist is given by: position coordinates (two real numbers), an orientation (0…360), the nest center and color of its respective colony. Note that an ant position corresponds always to a square cell. The perception of each ant is reduced to its own cell, but it is able to detect the nest center and move towards. The loaded virtual ants could wander around looking for their nests but moving in the nest direction, using the smallest length path, accelerates the painting process.

The behavior of *T. albipennis* sand artists is very simple: they collect building material into their colony nests. They depart from the center of their colonies, collect a single virtual grain of sand colored with their colony color and return to their virtual nests where they try to deposit it. Picking up and droppings are stochastic activities. The distance an ant is from the respective colony center is r. For an unloaded artist, the probability of picking up a virtual sand grain is P(r) and for a loaded artist the probability of dropping a piece of material is given by D(r). The quantities D(r) and P(r) are both influenced by the templates. For an artist, the function for dropping a virtual grain of sand will be maximum (D_M) at the circumference with radius ro_i ($r= ro_i$) (P(r) will be minimum at $r= ro_i$). The probability of picking up a virtual grain of sand is basically formula D(r) in 1 adapted to several colonies.

$$D(r) = \frac{D_M}{1+\tau(r-ro_i)^2}. \tag{1}$$

A smaller or larger τ value of corresponds to a wall with respectively, a smaller or larger width. We could easily consider different values for this parameter for different colonies in order to increase a higher variety of patterns but it is a global parameter. The probability of dropping a virtual grain of sand is also D(r) of [4] adapted to several colonies. The function for picking up a virtual sand grain is minimal when $r= ro_i$ and maximal (P_M) on the most distant points from the circumference with radius ro_i.

$$P(r) = P_M\left(1-\frac{1}{1+\tau(r-ro_i)^2}\right). \tag{2}$$

Both P_M and D_M are system parameters varying from 0 to 1. In fact, the values of these two parameters do not influence the global pattern but can slow down or accelerate the painting process. The behavior of our artificial artists is described in algorithm 1:

- ❖ An unloaded ant will wander around randomly. Whenever it sees a virtual grain of sand of its own colony it will pick it up with a probability P(r). If it succeeds it will be a loaded artist. Otherwise it continues to wander around randomly.
- ❖ A loaded ant will go directly towards its colony center if $r > ro_i$; it will wander around if $r \le ro_i$. It will try to drop its grain with probability D(r) whenever it is on a spot free of grains. When it succeeds it will be considered unloaded.

Algorithm 1. *T. albipennis* sand painter behavior

Wandering around means that an ant goes forward one step and rotate right some random degrees and then rotate left some random degrees. When the ants try to go forward outside the limits of the wall (a non toroidal canvas), they rotate randomly until they can go forward.

When loaded *T. albipennis* artists go towards their nest centers they will stay there until they are able to deposit the loaded grain. This means that if it is easier to deposit they will sooner start looking for grains again. Parameter τ plays an important role here because when it is small, the probability to drop is higher inside the nest center and vice-versa. But as the wall is built the cells with high dropping probability are progressively reduced. It means that after a while a lot of the loaded ants are insistently trying to drop their grains. Therefore, in each time step, with a bigger colony the global colony probability of dropping a grain is higher.

3.4 Grain Colors and Distribution

As we said before each colony will be associated with a color and thus we have only grains with colors that are associated with the existent colony colors. If we have four colonies that build magenta, blue, green and yellow walls, we will create sand painting material reduced to the colony colors. Circular walls with bigger center circles will need more grains but we distribute the grains in equal proportions for each color.

4 *T. albipennis* Art: Pointillistic Circular Walls

There are several parameters that we can vary: the number of colonies and consequent diversity of colors, the density and initial placement of grains placed on the canvas, the range of possible radius for the colony centers.

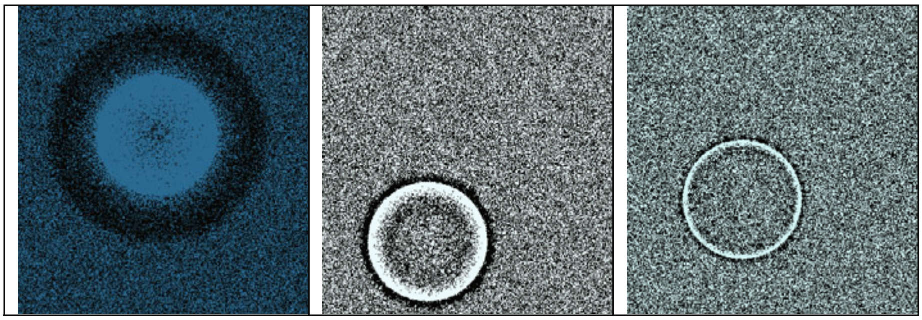


Fig. 3. Varying τ. We see 3 monochromatic sand paintings on a non toroidal canvas, built by a colony composed of 126 ants with the same r_0=76 and a density of grains of 49%. On the left painting τ=0,001, on the center τ=1,59 and on the right τ=27,7.

The painting begins by setting the parameters and then in each time step, all the ants, asynchronously, execute their behavior. There is no end for the sand painting, but even in face of competition between colonies, it usually converges to a stable pattern.

Let's build some sand paintings using just on colony to illustrate the wall building process and visualize the effects of parameter τ. In the paintings displayed on Fig. 3, we have disposed randomly some grains (density of 49%), which were crafted by a colony of 126 individuals. The center of the colony was randomly placed. It is clear the effect that τ has on the thickness of the corresponding wall.

In figure 4 we see 9 colored paintings, after playing with the values of relevant parameters. These paintings illustrate the diversity attained by the *T. albipennis* artists in spite of the existence of the same shape: the circle walls.

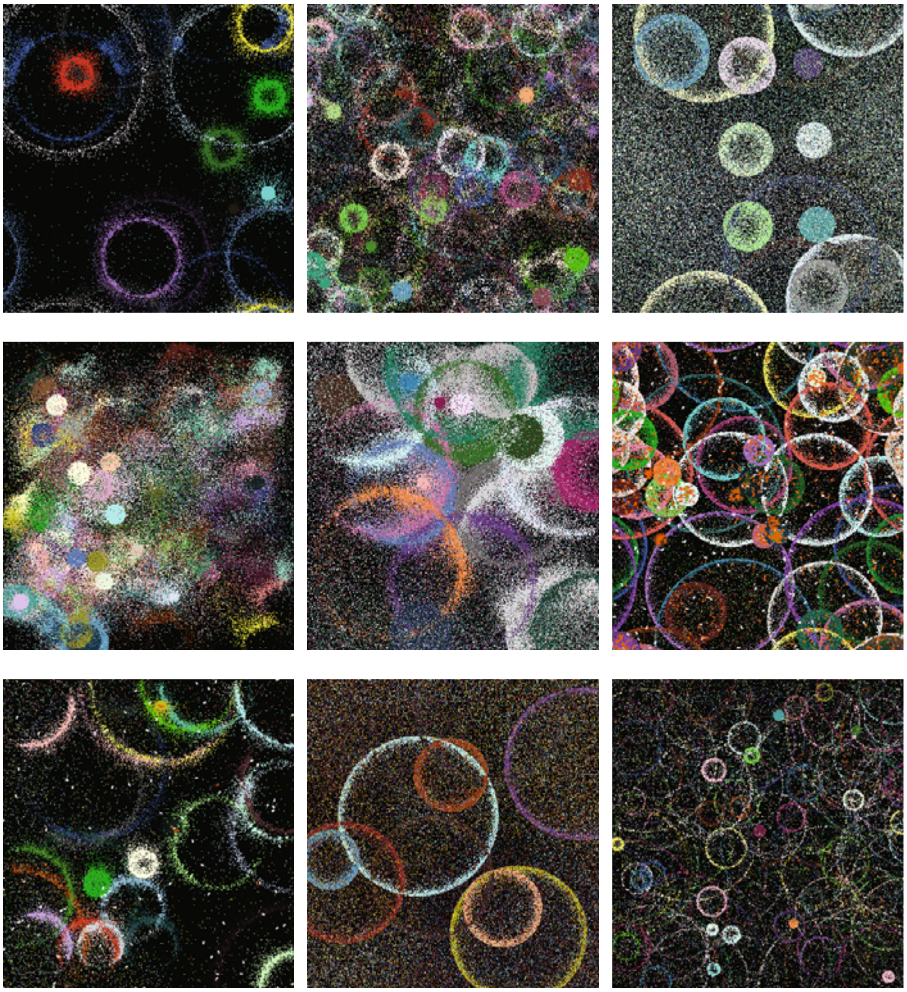


Fig. 4. Gallery of *T. albipennis* sand art. Nine sand paintings made from different parameter configurations where grains are disposed randomly on the canvas on a black background. In two of the paintings we can see the workers.

The different values for τ, the number of colors/colonies, the grain density and the random placement of nests along with their different center radius give us plenty of diversity. The duration of sand painting is dependent on the diverse parameters, which influence the process like P_M, D_M, population size and grain density— It is more difficult to find a free spot on a dense area. The density of grains plays also an important role on the global pattern, controlling the level of pointillism.

In order to expand pattern diversity we tried to change the initial distribution of virtual sand grains. We considered canvas where the grains on the left had much more grains than on the right or where the center of the canvas was more or less dense than the rest of the canvas space. The results are depicted in Fig. 5.

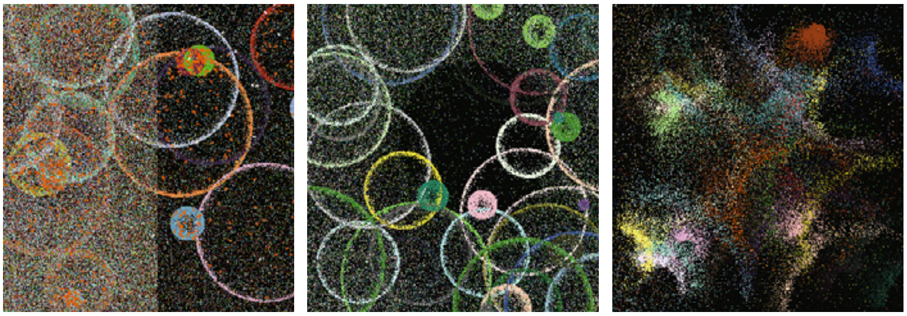


Fig. 5. Gallery of *T. albipennis* sand art. Six sand paintings made from different parameter configurations where grains are heterogeneously distributed on the canvas (black background). On the left painting we can see the sand painter artists, colored white (when they are unloaded) and orange (when they are loaded).

5 Future Work

The *T. albipennis* artists use only the template mechanism as the source for pattern formation. In the near future we will combine it with the amplification effect of "grains attracting grains". We know that *Temnothorax* ants exhibit a building behavior, which actively exploits existing heterogeneities in the building environment. For example, they can choose a nest cavity with pre-existing L shaped walls for their nest and close the wall, building an arc. Our artistic environment is more or less homogeneous and so this cannot happen but might be a future direction of our artistic research. We could use a new virtual material which cannot be picked up and start the painting process with a particular disposition of this material. We want to explore also the possibility of making walls with different shapes, beyond circles, using moving colonies that change nest site during the sand painting process and also colonies with no a priori template but where individuals negotiate their colony's template (nest center) in a collective decision process. Finally, we want to work with colonies with dynamic colony sizes so that the walls they build enlarge and reduce during the sand painting process.

References

1. Aupetit, S., Bordeau, V., Monmarché, N., Slimane, M., Venturini, G.: Interactive evolution of ant paintings. In: Sarker, R., Reynolds, R., Abbassss, H., Tan, K.C., McKay, B., Gedeon, T. (eds.) Proceedings of the 2003 Congress of Evolutionary Computation CEC 2003, pp. 1376–1383. IEEE Press, Los Alamitos (2003)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, Oxford (1999)
3. Boyd, J., Hushlak, G., Christian, J.: SwarmArt: Interactive Art from Swarm Intelligence. In: ACM Multimedia 2004, pp. 628–635. ACM, New York (2004)
4. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton Studies in Complexity (2001)

5. Franks, N.R., Wilby, A., Silverman, V.W., Tofts, C.: Self-organizing Nest Construction in Ants: Sophisticated Building by Blind Buldozing. Animal Behavior 44, 109–375 (1992)
6. Franks, N., Deneubourg, J.L.: Self-Organizing Nest Construction in Ants: Individual Behavior and Nest's Dynamics. Animal Behavior 54, 779–796 (1997)
7. Greenfield, G.R.: Evolutionary methods for ant colony paintings. 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. 478–487. Springer, Heidelberg (2005)
8. Greenfield, G.: Ant paintings using a multiple pheromone model. In: Sarhangi, R., Sharp, J. (eds.) BRIDGES 2006 Conference Proceedings, pp. 319–326 (2006)
9. Greenfield, G.: On Evolving Multi-Pheromone Ant Paintings. In: IEEE Congress on Evo. Comp., CEC, Vancouver, pp. 2072–2078. IEEE, Los Alamitos (2006)
10. Melhuish, C., Welby, J., Edwards, C.: Using templates for defensive wall building with autonomous mobile ant-like robots. In: Proceedings of Towards Intelligent Autonomous Mobile Robots 1999, Manchester, UK (1999)
11. Moura, L., Pereira, H.: Man + Robots: Symbiotic Art, Institut d'Art Contemporain, Lyon/Villeurbanne, France (2004)
12. Parker, C., Zhang, H., Kube, R.: Blind Bulldozing: multiple robot nest construction. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, USA (2003)
13. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Int. Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pp. 25–34. ACM, New York (1987)
14. Shiffman, D.: Swarm (2004), `http://www.shiffman.net/projects/swarm`
15. Urbano, P.: Playing in the Pheromone Playground: Experiences in Swarm Painting. 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. 527–532. Springer, Heidelberg (2005)
16. Urbano, P.: Consensual Paintings. 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. 622–632. Springer, Heidelberg (2006)

Merging Aesthetics with Functionality: An Interactive Genetic Algorithm Based on the Principle of Weighted Mutation

Eirini Vouliouri

11, Reas Street, 152 37 Filothei, Athens, Greece
eirini.vouliouri@gmail.com

Abstract. This paper proposes an algorithm through which the development of computationally generated forms can be externally directed towards both functional objectives and intuitive design targets. More precisely, it presents a special version of Interactive Genetic Algorithm, which introduces Weighted Mutation as a method to support the long life of genes corresponding to favored phenotypic characteristics. At the same time, optimization processes towards operational goals are also enabled. A set of experiments is conducted on the case study of a building façade, which appears to provide a suitable means for the investigation of functional, as well as aesthetic issues. The results are positively assessed, since they prove that this new methodology broadens the capacities of standard Interactive Genetic Algorithms, shedding light on how a constructive human-machine relationship can benefit the design process.

Keywords: Interactive Genetic Algorithm, Weighted Mutation, Human Evaluation, Algorithmic Design Control.

1 Introduction

The novelty brought forward by digital technologies raises a major issue regarding the notion of digital design control. Although technological progress has enriched the design domain with new innovative techniques, there is still an ongoing concern regarding the possible ways through which the algorithmic evolution of form can be directed towards a certain designer's particular goals. In this direction, Interactive Genetic Algorithms (IGAs) constitute a suitable means of externally guiding the computational development: By requesting the human evaluation as an indispensable step for the algorithm to proceed, an IGA structure favors the dynamic integration of the designer into the process of evolution.

However, the examples recorded so far refer to a sheer aesthetically-driven morphogenetic process, which does not involve any functionality restrictions. This fact challenges the contemporary research community to investigate new algorithmic structures, which will be capable of merging objective and subjective criteria into one single evolutionary procedure. This is also supported by Mutsuro Sasaki [7], who holds that it is exactly this unification of *mechanics* with *aesthetics* that will truly validate the generation of complex forms.

© Springer-Verlag Berlin Heidelberg 2011

This paper presents a variation of the conventional IGA structure, which deals with a more elaborate version within the operation of mutation: The proposed technique uses mutation *weighting*, as a method of favoring the survival of genes that control desired characteristics, without at the same time prohibiting any optimization processes towards functional goals. The experiments are carried out on the case study of a building façade, created through the use of L-Systems. It is suggested that this architectural element serves as an appropriate means for suchlike investigation, since it can be assessed both objectively, in terms of functionality, and subjectively, in terms of aesthetics.

The paper is structured as follows: The next section introduces the reader into the main principles of IGA and briefly outlines some characteristic examples. Section 3 reports on the experimentation, by explaining the actual methodology through which the façade is generated and the way it is combined with the Weighted Mutation IGA. For the conclusions to be meaningful, the outcomes are compared to the equivalent results of GA and standard IGA tests. A further development of the algorithm is proposed in Section 4, while Section 5 summarizes the presentation.

2 Interactive Genetic Algorithm

An Interactive Genetic Algorithm (IGA) is actually based on the structure of a standard Genetic Algorithm (GA), which follows the Darwinian theory of evolution. In a GA, the characteristics that specify a certain population of organisms are stored in a multitude of individuals, in the form of chromosomes. These are strings encoding problem solutions and consist of genes [4]. The total make-up of genes constitutes an individual's genotype, while the way the genotype is eventually expressed determines the phenotype of the individual.

Each organism is evaluated with respect to an objective scoring function. Subsequently, a new organism is created, by copying (crossover) and locally altering (mutation) the genotypes of two randomly selected individuals, which become the parents of the next generation. To accelerate the evolutionary process, the likelihood of selecting fitter individuals has to be higher than the probability of selecting poorly performing ones; yet it is crucial that a small probability is left for those less fit organisms to be selected, so that the population's characteristics do not converge early and the diversity of phenotypes is maintained during the initial stage of the algorithmic run. Once bred, the new organism is also evaluated and classified within the set of individuals, by excluding the less fit one. The whole process is repeated for a certain number of generations and the population is in this way directed towards the best performing solution.

GAs form robust searching techniques and are widely used for finding solutions to complex problems. Yet it is important that the size of the population is fairly large, so as to allow the searching of many test points within the search space of the algorithm. In this light, the operation of mutation contributes drastically to the testing of manifold gene combinations, by giving weak permutations the opportunity of being replaced by stronger ones, which, in another case, might have been lost.

In this direction, a typical IGA structure is not very different from a conventional GA methodology, though here it is the user that selects the individuals constituting the parents of the next generation. By repeating the algorithmic sequence for a considerable number of iterations, the best-performing design finally complies with the user's preferences. Astounding solutions can also be potentially generated, ones that could not have been achieved easily through alternative methodologies [8].

This idea of human intervention into evolutionary processes has its origins in Richard Dawkins' [3] seminal book "The Blind Watchmaker", where he describes a technique of user-computer collaboration, referenced as Interactive Evolution (IE). This approach has spawned various applications in the design field, such as the system invented by William Latham and Stephen Todd, which uses constructive solid geometry techniques to evolve "virtual sculptures" [5].

Referring to the IGA structure described above, a characteristic implementation concerns the "Drawing Evolver", a computer-aided drawing tool, developed by Ellie Baker and Margo Seltzer [1]. Here human evaluation is introduced dynamically into the algorithm, yet Baker and Seltzer notice that human speed and patience become limiting factors, restricting both the possible number of generations and the size of the population. As a solution, they suggest an increase of the mutation rate, which results in an expansion of the search space of the algorithm. Another IGA application is related to the fashion design domain [2]. Here, a variety of design details is encoded into chromosomes and the user, through constant interaction with the system, directs the way those items are recombined and reconfigured.

3 Presentation of the Experiments

The examples recorded concern the implementation of subjective and intuitive preferences into an algorithmic process, yet the fitness evaluation does not comprise any assessment in terms of explicitly defined functions. What the Weighted Mutation principle intends to define is an algorithmic methodology, capable of merging scientific and personal objectives as a single inherent property. This section firstly presents the algorithmic structure through which the architectural façade is generated, as well as the first test results, involving the façade's optimization towards lighting objectives by means of a Genetic Algorithm (GA). Then the structure is converted into an Interactive Genetic Algorithm (IGA), which enables the user's external contribution to the algorithmic process. The Weighted Mutation principle is subsequently examined as a third, separate case. All experiments are developed in Processing Programming Language.

3.1 L-Systems Generating an Architectural Façade

The pattern selected as the façade's main configuration is influenced by nature, particularly by the way plants and trees develop in time. The implementation of a façade in a GA for the purpose of lighting optimisation might be an issue already explored within the discipline of architecture [9]; however, this project doesn't treat the lighting parameter as a technical issue, but as a principle of morphogenesis.

The façade is generated through the use of L-Systems, which are algorithms simulating the development of multicellular organisms and plants [6]. Their structure is based on the technique of *rewriting*, referring to the generation of complex objects by successively replacing parts of a simple initial object; this is achieved through the use of rules, applied in a parallel process. In this particular project, L-Systems are not only employed because of the formalistic resemblance between natural forms and the produced patterns, but also because of the fact that small changes in the rewriting rules can result in significantly contrasting outcomes.

The façade's pattern consists of four branching units and is created according to the following procedure: At the beginning, four different polygons are placed at the corners of a rectangle, which has the same proportions as the façade. The orientation of each polygon is not set by the designer, but is randomly selected by the computer. This selection however is constrained by setting a maximum value for the angle formed by the central axis of each polygon and the associated side of the rectangle.

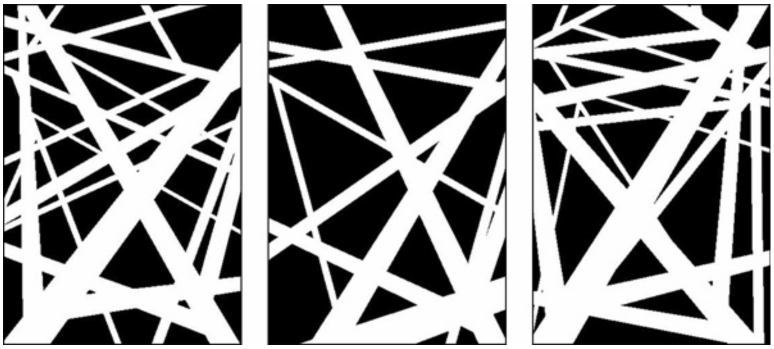


Fig. 1. Three different façade patterns deriving from the same algorithmic process

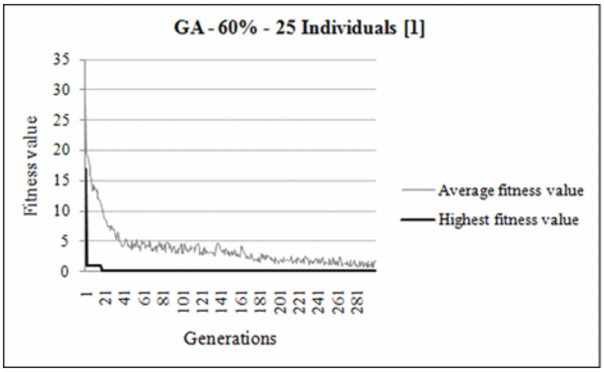
The polygons constitute initial components, *axioms* in the L-Systems vocabulary, which, at the next stage of the form development, are replaced by new ones, according to the applied rewriting rules. In this case, for each polygon "B", the substitution is defined by the rule "B $\rightarrow$ B (+B) (-B)", which means that every polygon is branching off two identical selves, one on each of its two sides. The new polygons are also subject to specific rules, such as the position of their pivot point, their rotation angle and a scaling factor which determines their proportion in relation to the previously generated polygon. The pivot points and the rotation angles are explicitly set; however, the scaling parameter is intentionally left to the computer to define so as to allow a random generation of thinner or thicker branches. The process is repeated by following the principle of recursion, which means that a function calls itself within its own block. To prevent this from continuing infinitely, a condition should be specified for the function to stop iterating. In this experiment, this is determined by the number of branching substitutions to be executed, which is also randomly selected by the computer within a certain range.

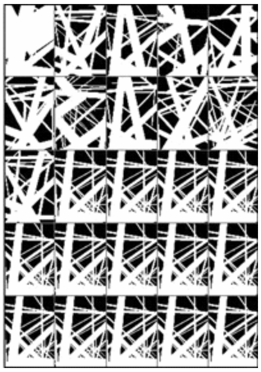
The simple L-System algorithm described above forms a principal process, called four times to generate the four units mentioned before. Figure 1 reveals that the same algorithm can result into different patterns. The L-System units form the solid part of the façade, while the remaining surface constitutes the façade's openings, allowing for natural light penetration.

3.2 Combination with a Genetic Algorithm for Lighting Optimization

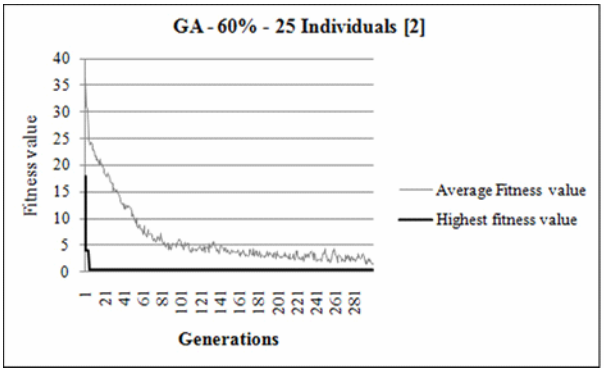
The L-System based algorithmic sequence is subsequently combined with a GA, which optimizes the façade in terms of the overall amount of light that penetrates it. If, for instance, the desired proportion of the openings on the whole surface of the façade is 60%, the algorithm will search for the best solution of creating a solid part of 40%. This can be achieved through different configurations, a fact which is positively assessed, as it will allow the user -in a further step- to select among different optimised results, according to his/her personal preferences.

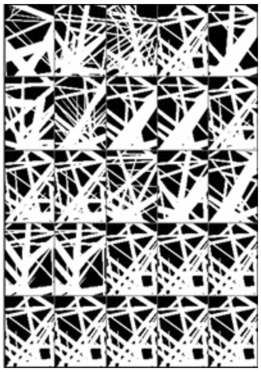
The evaluation function within the GA calculates the disparity between an individual's performance and the pre-specified goal, which means that the lower this value, the higher the fitness of the individual evaluated. Although a GA generally deals with large population sizes, here the experiments involve a population of only 25 individuals, due to the fact that the results will subsequently need to be compared with those deriving from the IGA experiments, which, as has already been mentioned, need to involve fewer individuals, so as not to cause human fatigue.





(a)





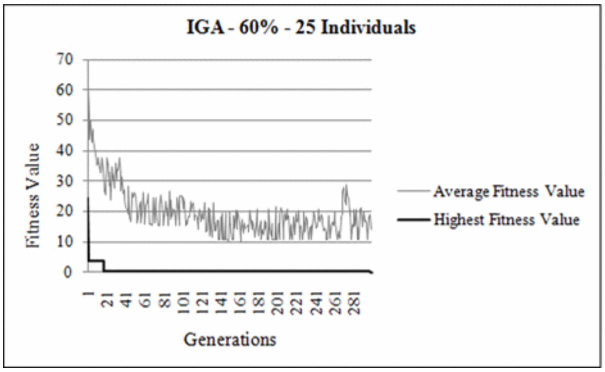
(b)

Fig. 2. Two GA Convergence Tests and Population Phenotypes

Regarding the experiments, two convergence tests were conducted, so as to test the similarity of the converging results, displayed on the right of each graph (Figures 2a and 2b). The algorithm ran for 300 iterations, using a mutation rate of 5%. The results show that elitist individuals, although performing almost equally in terms of functionality, may demonstrate a significant differentiation in the way the L-System branches are arranged, which reveals that the searching space of the algorithm contains multiple global optima. In this context, the challenge is to define an algorithmic process that will produce a variety of configurations in one single algorithmic process.

3.3 Conversion into an Interactive Genetic Algorithm

As a next step, the GA is converted into an IGA, with the user's task being to select the two individuals breeding the next generation. Once the parents are selected, the process follows the same structure as in the previous example. To make a fair comparison between the two experiments, the probability of mutation is intentionally kept at the same rate as before, that is 5%, and the population refers to a number of 25 individuals. Moreover, the algorithm runs for 300 iterations, having as a pre-specified target the value of 60%. Since such a procedure involves by definition the user's aesthetic preferences, these have to be taken into consideration for the evaluation of the results. In this particular experiment, the desired goal is characterised by a dense pattern, consisting of multiple, relatively thin branches.



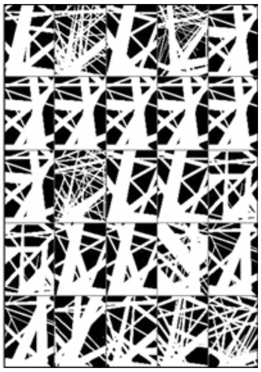
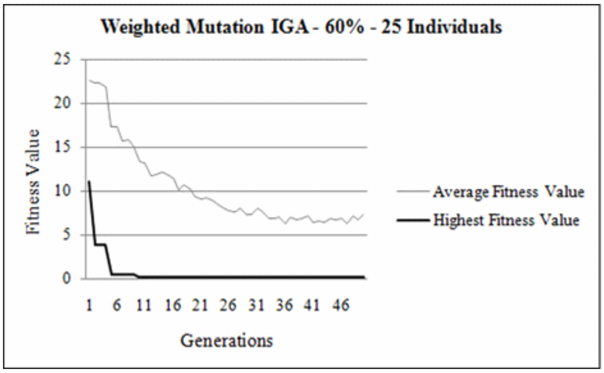


Fig. 3. IGA Convergence Test and Population Phenotypes

A comparison between the IGA experiment (Figure 3) and the GA tests reveals that the standard GA approach performs better in terms of functionality; yet it can be seen that the IGA phenotypes show a higher degree of variety, as they have not converged to one single solution; this is considered as positive. It can also be observed that the elitist individual (right bottom corner) meets the external user's expectations, as specified before, since it corresponds to a pattern that consists of multiple varying-thickness branches. On the contrary, the GA best performing individuals do not achieve this goal. However, the fact that the process has to be repeated for 300 iterations is negatively assessed, since it causes the user's burden.

3.4 The Weighted Mutation Principle

The Weighted Mutation Principle is presented as a structure through which the aforementioned problems can be overcome. It mainly involves a combination of GA and IGA methodologies: The user selects two individuals out of every new population; however, these do not constitute the parents of the next generation, but are rather stored in memory, forming the data to be used in a further stage of the evolution. Following the standard Genetic Algorithm (GA), two relatively fit individuals are randomly selected to become the parents of a new organism. The new genotype is subsequently compared to the user's selections: Each gene is compared to both of the corresponding genes of the externally chosen individuals. In case the new gene's value is different from both, its probability to mutate is increased to 10%, thus expanding the search space of the algorithm. If the value coincides with one of them, the mutation probability is reduced to 3.5%, while lastly, if it happens that all three values are identical, it becomes 1.5%. It is suggested that this methodology requires a smaller number of iterations to generate considerable results, since the mutation probability serves as a tool for a more efficient manipulation of the search space of the algorithm. Figure 4 illustrates the results of the experiment performed for a population of 25 individuals, which ran for only 50 iterations. The aesthetic goal is the same as before.



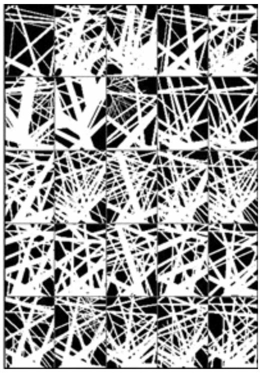


Fig. 4. Weighted Mutation IGA Convergence Test and Population Phenotypes

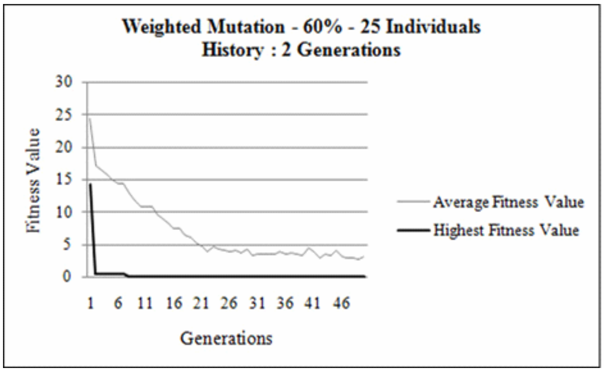
As can be seen from the graph, the average fitness value got stuck to a local optimum, after about 25 iterations. However, the phenotypes produced reveal that the user has actually managed to direct the algorithmic process towards the desired goal.

A more sophisticated methodology would take into consideration not only the last generation's selections, but also the ones indicated in the previous ones. In this direction, the mutation probability would be reduced according to the number of times a specific gene is selected by the user.

The following table shows the mutation probabilities for a history of one, two and three generations.

Table 1. Mutation probabilities used in history-based experiments

Number of generations	Genes Similarities		
	Two	**One**	**None**
1	1.5	3.5	10
2	1.0	3.0	10
3	0.5	2.5	10



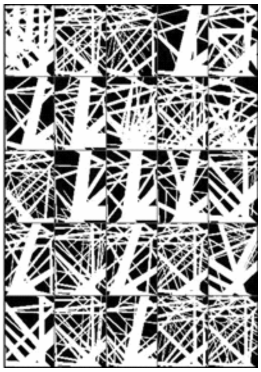
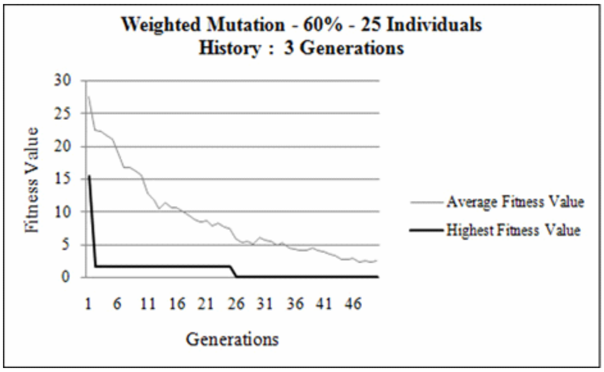


Fig. 5. 2-Generation History Weighted Mutation IGA Convergence Test and Population Phenotypes



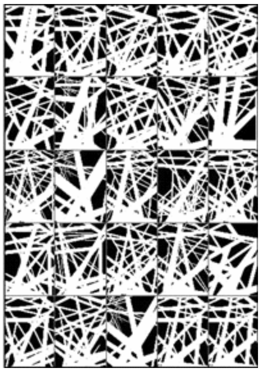


Fig. 6. 3-Generation History Weighted Mutation IGA Convergence Test and Population Phenotypes

Figures 5 and 6 illustrate the implementation of this process for two and three generations respectively. The results imply that the greater the number of generations taken into account, the better the process is guided towards both functional objectives and aesthetic goals. Although the average fitness values do not really converge to the optimum solution, which is the value of zero, they do perform significantly better than the ones of the standard IGA methodology.

4 Future Work

A more sophisticated version of the algorithm would concern the method of comparing the genes of the individuals selected by the user to the one selected by the computer. In the algorithm presented before, this comparison involves a binary result, meaning that if the two numbers compared are slightly different, they will still be marked as not equal. This method can be enhanced so that the probability of mutation does not depend on the equality or non-equality of the numbers but on the difference or the ratio of those two. The user could then specify the values of the mutation probability according to the comparison results.

Moreover, further experiments could involve multiple users assessing the process with respect to whether they have managed to get the desired results. The effectiveness of the proposed methodology could also be tested through the combination of different algorithms with a Weighted Mutation IGA.

5 Conclusions

This paper has been built on the basis of an Interactive Genetic Algorithm, which utilizes Weighted Mutation as a technique of guiding the evolution of form towards objective, as well as subjective criteria. In this structure, the operation of mutation, instead of performing in a blind, deterministic manner, contributes drastically to the user's personal manipulation of the search space of the algorithm. The results, carried out on the case study of a building façade, have been positively assessed, since they proved that the capacities of the standard Interactive Genetic Algorithm can thus be broadened. Apparently, the proposed algorithmic structure extends beyond the boundaries of this particular project, by elucidating ways through which the design process can be enhanced.

Acknowledgments. The project has been developed as part of postgraduate studies in MSc Adaptive Architecture and Computation - University College London.

References

1. Baker, E., Seltzer, M.: Evolving Line Drawings. In: Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 91–100. Morgan Kaufmann Publishers, San Francisco (1994)
2. Cho, S.-B.: Towards Creative Evolutionary Systems with Interactive Genetic Algorithm. In: Applied Intelligence, vol. 16, pp. 129–138. Kluwer Academic Publishers, Dordrecht (2002)
3. Dawkins, R.: The Blind Watchmaker. Longman Scientific and Technical, Harlow (1986)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
5. Latham, W., Todd, S.: Evolutionary Arts and Computers. Academic Press Inc., Orlando (1994)
6. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, New York (1978)

7. Sasaki, M.: Morphogenesis of Flux Structure. From the realization of Free 3D Forms by Architects to Evolving the Most Logical Structure by Computer Calculation. In: Sakamoto, T., Ferré, A.T. (eds.) From Control to Design. Parametric/Algorithmic Architecture, pp. 68–69. Actar, Barcelona (2007)
8. Sims, K.: Artificial Evolution for Computer Graphics. Computer Graphics 25(4), 319–328 (1991)
9. Torres, S.L., Sakamoto, Y.: Façade Design Optimization for Daylight with a Simple Genetic Algorithm. In: International Building Performance Simulation Association, pp. 1162–1167 (2007), http://www.ibpsa.org/proceedings/BS2007/p117_final.pdf

Nature-Inspired Optimization for Biped Robot Locomotion and Gait Planning

Shahriar Asta and Sanem Sariel-Talay

Computer Engineering Department
Istanbul Technical University, Istanbul, Turkey
{asta,sariel}@itu.edu.tr

Abstract. Biped locomotion for humanoid robots is a challenging problem that has come into prominence in recent years. As the degrees of freedom of a humanoid robot approaches to that of humans, the need for a better, flexible and robust maneuverability becomes inevitable for real or realistic environments. This paper presents new motion types for a humanoid robot in coronal plane on the basis of Partial Fourier Series model. To the best of our knowledge, this is the first time that omni-directionality has been achieved for this motion model. Three different nature-inspired optimization algorithms have been used to improve the gait quality by tuning the parameters of the proposed model. It has been empirically shown that the trajectories of the two specific motion types, namely side walk and diagonal walk, can be successfully improved by using these optimization methods. The best results are obtained by the Simulated Annealing procedure with restarting.

Keywords: Biped robot locomotion, gait planning, coronal plane, sagittal plane, omnidirectional locomotion, simulated annealing, genetic algorithms.

1 Introduction

Gait planning is essential for a robot to navigate in an environment. Especially humanoid robot gait planning problem has come into prominence due to the recent research trend toward human-like robots and the difficulty of generating stable motions for biped locomotion. Several different gait planning models have been developed for humanoid robots. These models rely on various mathematical and kinematic approaches. ZMP based motion model which was originally introduced by Vukobratović and further developed and extended by others [1-3], is one of the most popular gait generation models in mobile robotics. In this approach, the main objective is to design robot's motion in such a way that the zero moment point (the point where total inertia force equals to zero), does not exceed a predefined stability region. Another popular gait generation model considers the Center Of Mass (COM) to evaluate the quality of the generated gait [4]. Central Pattern Generator (CPG) method [5] generates gaits by considering a lattice of neurons for the production of repetitive patterns.

As a recent work, the Partial Fourier Series (PFS) method [6] was introduced for generating a forward walk motion in the sagittal plane. The parameters of the proposed

© Springer-Verlag Berlin Heidelberg 2011

model were optimized using a Genetic Algorithm. The PFS method is easy to implement on humanoid robots, and promising results has been observed for a forward walk motion in [6].

We extend the work by [6] by applying the PFS model for different types of biped motions. The main objective of this paper is to introduce omni-directionality to PFS motion model. The first step toward this goal is to show that this motion model is suitable for biped locomotion in coronal plane in addition to the sagittal plane. For that purpose, the models of two important motions for a humanoid robot, namely, side walk and diagonal walk are introduced in this plane. These motions are constructed based on the physical and kinematic models. However, their parameters are needed to be optimized. Therefore, an optimization step is needed. In this paper, the parameters of PFS model are optimized by using nature-inspired methods: variations of Simulated Annealing and Genetic Algorithms. This paper presents both the optimized parameters and a comparative analysis of these methods for humanoid gait planning problem.

The rest of the paper is organized as follows. First, we present a brief review on the humanoid robot mode used in the experiments and its basic motion model. Then, the selected Simulated Annealing and Genetic Algorithms are reviewed. After this review, the two new motion types in coronal plane are introduced. It has been shown that the robot is able to perform a side walk and a diagonal walk using the Partial Fourier Series model. After presenting the model, the optimization processes are discussed. Finally, the experiments and the results are presented followed by the conclusion.

2 Motion Model

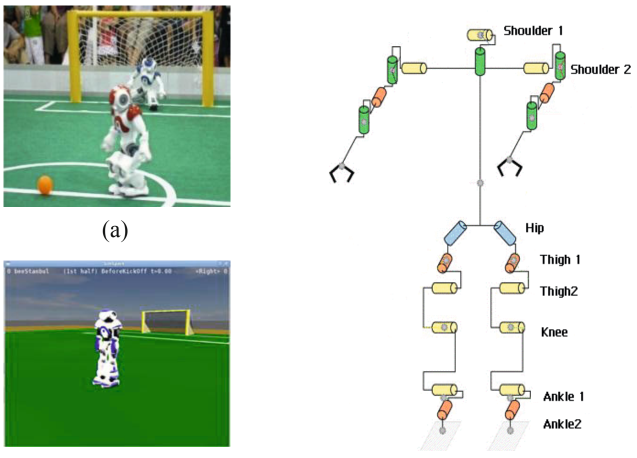
The PFS model was first introduced by [6] for the simulated Aldebaran Nao robot model in Simspark simulation environment. Simspark is the official simulator for RoboCup competitions and uses ODE (Open Dynamics Engine) for physics simulation [7]. We use the same robot model for presenting our proposed motions.

Nao robot has 22 degrees of freedom of which only 12 have been used in this motion model. The height of the robot is 57 cm and its weight is 4.3 kg. Since the simulated Nao robot (Fig. 1(b)) is a realistic representation of the real Nao humanoid robot, its joint structure (Fig. 1(c)) is the same with the real one.

The human walking motion can be modeled by a smoothed rolling polygonal shape and a periodic function accordingly. In this sense, one can use the principle of Partial Fourier Series (PFS) in order to decompose the bipedal walk's periodic function into a set of oscillators [9]. Assigning these oscillators to the joints of a humanoid robot enables one to develop a gait for the robot. According to [4], the main periodic function for the bipedal walk can be formulated as following:

$$f(t) = C + \sum_{i=1}^{N} A_i \sin(i\frac{2\pi}{T}t + \emptyset_i) \tag{1}$$

Where N is the number of frequencies (degrees of freedom which are used in gait definition), C is the offset, $A_{1 \dots N}$ are amplitudes, T is the period and $\emptyset_{1 \dots N}$ are phases.



(a)

(b) (c)

Fig. 1. a) The Aldebaran Nao robot at RoboCup Standard Platform League [8] b) Simulated Nao Robot at Simspark environment, c) Nao Robot Structure [7]

This PFS model ensures the right and left feet of the robot alternately perform swing and support roles. Note that, this alternation can be achieved by adding a shift of π for pitch joints of the right foot. The above mentioned oscillators give the desired angle of joints in a specific time. In order to control the robot with these joint angles, we use a simple proportional control method:

$$Speed * (\theta_{Target} - \theta_{Current}) \tag{2}$$

Sending the outcome to the robot's motion interface and then to Simspark causes the joints to move with the desired speed and value.

3 Motions in Coronal Plane

There are three major planes in which humanoid robot locomotion is considered: Sagittal plane, Coronal plane and Transverse plane. Forward and backward walk motions moves the robot in the sagittal plane. Motions in Coronal plane move the robot toward the sides. And finally, Transverse plane motions change the orientation of the robot. In this paper, we have extended the PFS model which is proposed for forward walk [6] into a model for coronal plane motions: side walk and diagonal walk.

3.1 Side Walk

The side walk motion is a coronal plane motion which does not change the orientation of the robot. The side walk motion involves the oscillation of two types of joints, namely, the pitch and the roll joints. We use the pitch joints to produce single support locomotion. This is based on a simple intuition rather than a detailed kinematics analysis. The idea is closing the swinging foot's pitch joints, that is, assigning values to these joints so that the foot is detached from the ground and moved toward the robot's torso. Opening pitch joints has exactly the reverse meaning. Closing pitch joints opens space for the swinging foot to move along the coronal plane and prevents from collision with the ground. At the same time, the support foot's pitch joints are opened to expand the space created for the swinging foot. For roll joints however, opening means, assigning values to the joints so that the foot moves away from the torso side ward and closing means the reverse action.

Both feet's roll joints have to be opened to opposite sides simultaneously to their respective leg's pitch joints. When the swinging foot touches the ground, the pitch joints should be close to their offset value to ensure sideward stability. This is where the roll joints have to be at their maximum values in order to maximize the force applied to the ground. In other words, there should always be a $\pi/2$ phase difference between the roll and pitch joints of the same leg. At the same time, in order to ensure that roll joints are closed and opened in opposite directions there should also be a phase difference of π between the corresponding roll joints of the two legs. Note that, in order to maintain a support-swing sequence, a phase difference of π is applied between the corresponding pitch joints of the two legs.

Table 1. The joint oscillators for a side walk motion in the coronal plane

Left Leg Joints	Right Leg Joints
$f_{LShoulder2}(t) = C_1 + A_1 \sin\left(\frac{2\pi t}{T} + \frac{\pi}{2} + \varphi_1\right)$	$f_{RShoulder2}(t) = -C_1 + A_1 \sin\left(\frac{2\pi t}{T} - \pi/2 + \varphi_1\right)$
$f_{LThigh1}(t) = C_2 + A_2 \sin\left(\frac{2\pi t}{T} + \pi/2 + \varphi_2\right)$	$f_{RThigh1}(t) = -C_2 + A_2 \sin\left(\frac{2\pi t}{T} - \pi/2 + \varphi_2\right)$
$f_{LThigh2}(t) = C_3 + A_3 \sin\left(\frac{2\pi t}{T} + \varphi_3\right)^-$	$f_{RThigh2}(t) = C_3 + A_3 \sin\left(\frac{2\pi t}{T} + \pi + \varphi_3\right)^-$
$f_{LKnee}(t) = C_4 + A_4 \sin\left(\frac{2\pi t}{T} + \pi + \varphi_4\right)^+$	$f_{RKnee}(t) = C_4 + A_4 \sin\left(\frac{2\pi t}{T} + \varphi_4\right)^+$
$f_{LAnkle1}(t) = C_5 + A_5 \sin\left(\frac{2\pi t}{T} + \varphi_5\right)^-$	$f_{RAnkle1}(t) = C_5 + A_5 \sin\left(\frac{2\pi t}{T} + \pi + \varphi_5\right)^-$
$f_{LAnkle2}(t) = C_6 + A_6 \sin\left(\frac{2\pi t}{T} + \pi/2 + \varphi_6\right)$	$f_{RAnkle2}(t) = -C_6 + A_6 \sin\left(\frac{2\pi t}{T} - \frac{\pi}{2} + \varphi_6\right)$

The set of equations in Table 1 presents the coronal plane motion gait planning equations for each joint. One has to note that, in the kinematic design of the simulated Nao robot, the roll joints' y axes (horizontal) point to the opposite directions [7]. That is the reason for using negative coefficients for *RShoulder2*, *RThigh1* and *Rankle2* joints. The offset and amplitude values of the corresponding *Thigh2* and *Ankle2* joints are initially set as the same for the simplicity of the model ($A_3 = A_5$,$C_3 = C_5$). The + and – superscripts in the formulation of some joints represent positive or negative half of their sinusoids respectively. The formulation in Table 1 results in a side walk motion with an average speed at 40 cm/sec. However, the parameters are needed to be optimized for a faster and straight sidewalk.

3.2 Diagonal Walk

Diagonal walk is another useful motion in the coronal plane. The robot is supposed to plan a straight trajectory line at a 45 degree angle with the horizon, without changing its orientation. The same intuition for the side walk motion also applies to the diagonal walk. One has to note that, the diagonal walk motion is a variation of a side walk motion for which both forward and sideward force vectors are applied. This is why an additional joint, the hip joint in Fig1(c), is used to create the required forward force. Therefore, the diagonal walk motion model can be given as an extended side walk model with the following two PFS formulations (one for each leg's hip joint).

$$f_{LHip}(t) = -C_{Hip} + A_{LHip}\sin\left(\frac{2\pi t}{T} + \pi/2 + \varphi_{Hip}\right)$$
$$f_{RHip}(t) = -C_{Hip} + A_{RHip}\sin\left(\frac{2\pi t}{T} - \pi/2 + \varphi_{Hip}\right) \quad (3)$$

Note that, unlike other joints, amplitudes for the right and left hips are not necessarily the same. Just like side walk, we assign parameters to C_{Hip}, A_{LHip} and A_{RHip} based on a simple reasoning. The resulting diagonal walk motion does not follow the desired trajectory. Snapshots of these initial side walk and diagonal walk motions are shown in Fig. 2.

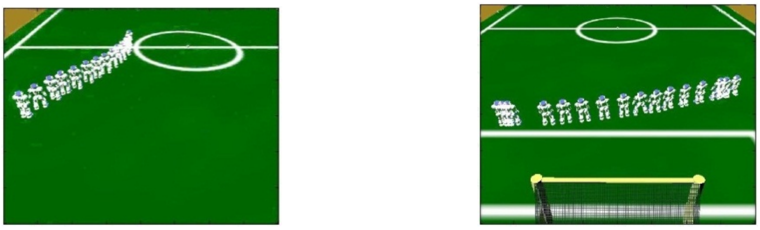

(a) (b)

Fig. 2. The motion patterns before optimization (a) side walk (b) diagonal walk

4 Optimization Methods

We have investigated Simulated Annealing and Genetic Algorithm optimization methods in order to optimize the parameters of the motion model of a humanoid robot.

Simulated Annealing (SA) is a global optimization method which applies a variation of Hill Climbing search to improve the solution quality but performing probabilistic random moves to escape from local optima. The probability value decreases exponentially according to a function of the temperature (*T*) value as in the original annealing process [10]. At higher values of *T* bad moves are more likely to be allowed. As *T* closes to zero, bad moves are less likely to happen, until *T* is zero in which case, the algorithm behaves like Hill Climbing. In this paper, we have tested two configurations of Simulated Annealing procedure. One configuration is the simple SA algorithm as mentioned above. The other includes restarting after a cooling procedure.

In SA with restarting (SAR), the simulated annealing algorithm is iterated inside a main loop which repeatedly resets the temperature to its initial value. A small cooling factor which causes sharp temperature decrease after the evaluation of each individual is used in the annealing process. This reduces the number of annealing iterations and the number of times that SAR considers to replace the current individual with the next one. In such a case, the probability of replacing the current individual is less than that of SA without restart.

```
Initialize (Curr , CurrentFitness = Fitness(Curr) , T)
for (iteration) do
    reset temperature
    while (T >= End Temperature)
        Next ← ComputeNext ()
        $\Delta E$ ← Fitness(Next) – CurrentFitness
        if($\Delta E$ < 0)
            Curr ← Next
            CurrentFitness ← CurrentFitness + $\Delta E$
        else
            $p = e^{-\Delta E/T}$
            if(p > 0.5)
                Curr ← Next
                CurrentFitness ← CurrentFitness + $\Delta E$
        if (Fitness < $\Delta E$)
                CurrentFitness ← Fitness
        T ← T * CoolingRate
```

Fig. 3. Pseudo code for the simulated annealing with random restart

In Genetic Algorithms (GAs), a population of individuals is evolved through generations. Each individual contains a possible solution to the problem in the form of numerical values. In order to generate the next population, individuals undergo several operations [10]. There are three major operations in a simple Genetic Algorithm: selection, crossover and mutation. A standard genetic algorithm approach is used in this research. The corresponding parameters for these optimization methods are presented in the next section.

5 Optimization of Motion Models

As discussed in the previous sections, the constructed PFS model produces the specified motion patterns. However, the parameters are needed to be tuned to obtain the motion trajectories in their desired sense (e.g., the speed of the side walk motion needs to be fastened and the orientation should be kept constant during a diagonal walk). Nature-inspired optimization methods are suitable for addressing these types of problems due to their size and complexity. In this work, three optimization methods, namely, Simulated Annealing (SA), Simulated Annealing with restarts (SAR) and Genetic Algorithms (GA) have been utilized to optimize the parameters and a comparative analysis is provided for humanoid gait planning problem. The following three subsections present the settings of the algorithms. Different fitness functions are constructed for each motion type.

The fitness function for the side walk is expressed as :

$$fitness = d_{target} + \bar{\theta}, \quad \bar{\theta} = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\bar{x})+\sum_{i=1}^{N}(y_i-\bar{y})+\sum_{i=1}^{N}(z-\bar{z})}{N}} \tag{4}$$

This fitness function has been adapted from [6]. Here d_{target} is the final distance to the target at the end of each simulation trial. The target is fixed and defined as the location of the ball at the center of the field. $\bar{\theta}$ is the average oscillation of the torso (in radians per second). The oscillation values are received from robot's gyro which is installed on the torso. Note that, the Simspark is a realistic simulation environment in which actions are nondeterministic and noisy. That is, the outcome of an action is not completely predictable as desired. Therefore, the fitness value of a parameter set is determined as an average of 3 simulation trials.

Similarly the fitness function for the diagonal walk is designed as :

$$fitness = d_{target} + \bar{\varphi} \tag{5}$$

Where d_{target} is the distance to the target from the initial location of the robot. The target is simulated as the fixed location of the ball at the center of the field. $\bar{\varphi}$ is the average absolute value of the horizontal angle detected by the robot's camera during a trial.

$$\bar{\varphi} = \frac{\sum_t |\varphi_t|}{N} \tag{6}$$

In both equations (4) and (6), *N* is the number of simulation cycles in each trial. A simulation cycle is 0.02 sec. Duration of each trial is 10 and 7 seconds for the diagonal walk and the side walk respectively. The initial location of the robot at the start time of a trial is (-8,0) for side walk and (-5,5) for diagonal walk. For the side walk experiments, the robot's orientation is set to an angle such that torso's x axis is parallel to the midfield line. At the start of each trial, the robot's head is rotated 90 and 45 degrees toward the target(i.e., the ball) for the side walk and diagonal walk, respectively. Corresponding angle for the head is kept constant during a trial.

The selected methods and the parameter values for GA, SA and SAR during the optimization of side walk and diagonal walk gait planning are shown in Table 2.

Table 2. The experimental settings for the optimization algorithms

Genetic Algorithm				
$P_{mutation}$	**$P_{crossover}$**	**Crossover Type**	**Population Size**	**Stop Criteria**
0.55	0.85	Uniform	50	Similarity of population
Elitism	**Selection Policy**	**Mutation Type**	**Fitness Function**	**# of Parameters**
10%	Proportional	Random from the range of joint values	Minimization	8 (SW)
Simulated Annealing				
Initial Temperature	**Cooling Rate**	**End Temperature**	**Fitness Function**	**# of Parameters**
4000	0.99	0.001	Minimization	11 (DW)
Simulated Annealing with Restart				
Initial Temperature	**Cooling Rate**	**End Temperature**	**Fitness Function**	**# of Parameters**
400	0.5	0.001	Minimization	11(DW)/8(SW)

In GA, each individual represents the parameters $\boldsymbol{\varphi_1} \dots \boldsymbol{\varphi_6}$and T in equations of table 1 along with the speed in Equation (2). All the represented values in an indivial are double. SA in its original form has been used for diagonal walk optimization. Each individual in SA represents the parameters $\boldsymbol{\varphi_1} \dots \boldsymbol{\varphi_6}$ and T in equations of table 1, and the $\boldsymbol{\varphi_{Hip}}$, $\boldsymbol{A_{LHip}}$, $\boldsymbol{A_{RHip}}$, $\boldsymbol{C_{Hip}}$ in Equation (3). Note that, diagonal walk has 3 more parameters as compared to side walk , while speed is not considered as a parameter for Diagonal Walk. SAR has been tested and applied on both motions with the same setting. In SAR, the cooling rate is set to a lower value than that of the simple SA and the annealing is performed fewer times (20 times).

6 Experimental Results

The selected optimization algorithms have been tested for optimizing the parameters for the sidewalk and diagonal walk motions. Each algorithm on each respected motion type was run for 10 times. Since the simulation environment is non-deterministic, the fitness value for each individual is determined as an average of 3 evaluations.

Table 3. The overall best fitness values averaged over 10 runs for the two motion models

Side walk				Diagonal Walk			
	μ	σ	Confidence Interval (99%)		μ	σ	Confidence Interval (99%)
GA	7.14	0.35	6.79 - 7.49	SA	3.17	0.18	2.99 - 3.35
SAR	5.83	0.39	5.44 – 6.15	SAR	2.29	0.13	2.16 - 2.42

The overall results (with the 99% confidence interval) are presented in Table.3. As can be seen, SAR performs better than SA and GA. It results in a lower mean fitness value among 10 runs and there is no overlap between the interval values.

SAR outperforms GA for the side walk motion. However at some points, the SAR shows a bad performance which closes it to the performance of GA. One other interesting fact is that sometimes the GA shows a better performance than its mean performance and closes down to the SAR's worst performance. This is due to the random nature of GA which is much more emphasized than SAR

A single run's comparative result is presented in Fig4 (a). As can be seen from these results, SAR can find the improved parameters in less than 250 iterations for the diagonal walk in one single run. This is equivalent to running the simple SA algorithm with the parameters in Table 2 for 10 loops with restarts. This comparative study illustrates that SAR outperforms the simple SA which is not able to find a reasonable result even in more iterations. When the performance of SAR is analyzed for two different motions, the convergence to the improved parameters for the diagonal walk has been observed in later iterations compared to that of the side walk. This is due to the structure of the diagonal walk which uses an additional joint (hip joint).

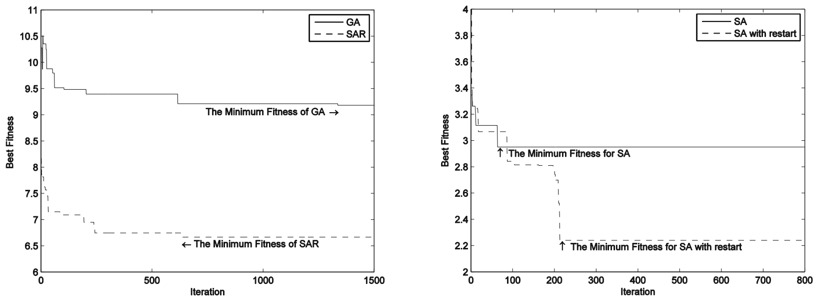


Fig. 4. Single run results of (a) GA, SAR for side walk. (b) SA, SAR for diagonal walk.

The improved side walk motion (resulting from SAR optimization) has a speed of 55 cm/sec (which is slightly better than its original version at 40 cm/sec) and the deviation from a straight baseline is considerably small. The diagonal walk motion with the optimized parameters ensures the robot to keep its orientation at a 45 degree straight line toward the target, and its speed is 35 cm/sec. The snapshots of the resulting motions are shown in Fig 5.

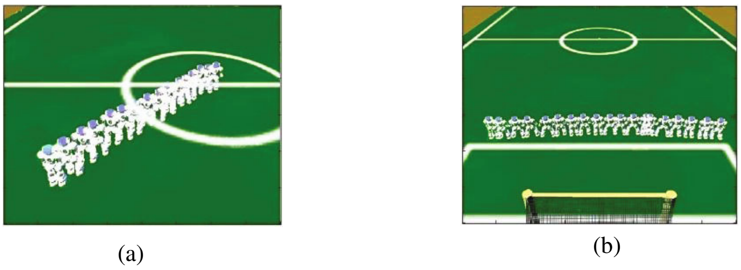


Fig. 5. (a) improved diagonal walk (b) improved side walk

7 Conclusions

In this work, we have shown that the PFS model can be successfully extended for omni-directional motions on humanoid robots. However, an optimization step is needed to specify the tuned parameters for the PFS model for different motions. It has been empirically shown that the optimization can be performed by using nature-inspired optimization techniques. A comparative analysis of three optimization methods is given for gait planning of two motion models for the simulated Nao robot in Simspark environment. As the experimental results illustrate, the best parameters are obtained by using Simulated Annealing with restarts. These best parameters produce the desired motion trajectories. The future work includes a general omni-directional gait planning based on the introduced PFS model for all motion directions.

Acknowledgement. Authors would like to thank Ismail Hakki Yadigar for his contribution in the implementation of the nature-inspired methods.

References

1. Vukobratović, M., Borovac, B.: Zero-moment point—Thirty five years of its life. International Journal of Humanoid Robotics 1(1), 157–173 (2004)
2. Yanjun, Z.: Fuzzy Omnidirectional Walking Controller for the Humanoid Soccer Robot. In: IEEE-RAS International Conference on Humanoid Robots (2009)
3. Gao, Z., Wu, J., Chen, H., Pan, B., Zheng, H., Liang, D., Liu, X.: Omnidirectional Motion Control for the Humanoid Soccer Robot. In: Cai, Z., Li, Z., Kang, Z., Liu, Y. (eds.) ISICA 2009. CCIS, vol. 51, pp. 1–8. Springer, Heidelberg (2009)
4. Graf, C., Härtl, A., Röfer, T., Laue, T.: A Robust Closed-Loop Gait for the Standard Platform League Humanoid. In: IEEE-RAS International Conference on Humanoid Robots (2009)
5. Carla, M.A.P., Golubitsky, M.: Central Pattern Generators for Bipedal Locomotion. Journal of Mathematical Biology 53(3), 474–489 (2006)
6. Picado, H., Gestal, M., Lau, N., Reis, L.P., Tome, A.M.: Automatic Generation of Biped Walk Behavior Using Genetic Algorithms. In: Proceedings of the 10th International Conf. on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence (2009)
7. Boedecker, J., Dorer, K., Rollman, M., Xu, Y., Xue, F., Buchta, M., Vatankhah, H.: Simspark Manual (2010)
8. Aldebaran Official Web Site, `http://www.aldebaran-robotics.com/`
9. Shafii, N., Javadi, M.H., Kimiaghalam, B.: A Truncated Fourier Series with Genetic Algorithm for the control of Biped Locomotion. In: Proceeding of the 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1781–1785 (2009)
10. Russell, S.J., Norvig, P.: Artificial Intelligence, A Modern Approach, 2nd edn. (2003)

Experimental Comparison of Selection Hyper-heuristics for the Short-Term Electrical Power Generation Scheduling Problem

Argun Berberoğlu[1] and A. Şima Uyar[2]

[1] Istanbul Technical University, Informatics Institute
[2] Istanbul Technical University, Department of Computer Engineering, 34490, Maslak, Turkey
{aberberoglu,etaner}@itu.edu.tr

Abstract. This paper presents an experimental comparison of a selection hyper-heuristic approach with several heuristic selection and move acceptance strategy combinations for the Short-Term Electrical Power Generation Scheduling problem. Tests are performed to analyze the efficiency of the combinations using problem instances taken from literature. Results show that the hyper-heuristic using the random permutation descent heuristic selection method and the only improving move acceptance scheme achieves the best results on the chosen problem instances. Because of the promising results, research will continue for further enhancements.

Keywords: Short-term electrical power generation scheduling problem, hyper-heuristics, unit commitment problem, optimization, scheduling.

1 Introduction

The short-term electrical power generation scheduling (SEPGS) problem is a constrained optimization problem, which aims at selecting operating units and determining the working hours of the units to produce power at a minimum cost while providing the hourly forecasted power demand without violating any constraint.

In literature, this scheduling problem is also known as the unit commitment problem and it attracts great interest in the business world as well as in the academic world, since optimal scheduling decreases the power generation costs significantly. Due to this, many optimization techniques, such as dynamic-programming [1], Lagrangian relaxation [2], tabu search [3], simulated annealing [4], branch and bound [5], priority lists [6], greedy algorithms [7], ant colony optimization [8] and evolutionary algorithms [9,10,11,12,13] have been applied to the SEPGS problem.

In our previous study [13], the performance of a hyper-heuristic approach with the random permutation descent heuristic selection method and the only improving move acceptance scheme is compared with other optimization techniques found in literature on four commonly used benchmark data sets. This paper explores the various heuristic selection and move acceptance method applications to the SEPGS problem to determine the best combination.

© Springer-Verlag Berlin Heidelberg 2011

2 Short-Term Electrical Power Generation Scheduling Problem

The objective of the SEPGS problem is to minimize the power generation costs over a given time period. Power generation costs include fuel costs, start-up costs and maintenance costs. Penalty coefficients are used to handle infeasible candidate solutions. These penalty values are the demand penalty, which occurs when the predefined hourly power demand is not fulfilled by the candidate solution, and the up/down penalty, which is added to the objective function due to additional maintenance costs, when an up/down constraint is violated for at least one generator. The following parameters are used in the SEPGS formulation.

Table 1. Parameters used in the definition of the SEPGS

Parameter	Explanation
$P_i(t)$	generated power by unit *i* at time *t*
$F_i(p)$	cost of producing *p* MW power by unit *i*
PD(t)	power demand at time *t*
PR(t)	power reserve at time *t*
$CS_i(t)$	start-up cost of unit *i* at time *t*
$x_i(t)$	duration for which unit *i* has stayed online/offline since hour *t*
$v_i(t)$	status of unit *i* at time *t* (on-off)

Turning a power unit on, brings an additional cost, which is called the start-up cost. The start-up cost depends on the type of the corresponding power unit and the amount of time the unit has stayed offline before starting to operate. This cost is calculated as shown in Eq. 1.

$$CS_i(t) = \begin{cases} CS_{hot} & if \quad x_i(t) \le t_{coldstart} \\ CS_{cold} & otherwise \end{cases} \tag{1}$$

where $t_{coldstart}$ defines the number of hours that it takes for the generator to cool down. This value is used as the threshold to determine, whether the start-up is a cold or a hot start-up depending on the generator type. The cost of a hot start-up is higher than the cost of a cold start-up.

The minimum up/down operational constraint defines the minimum up-time t_{up} and the minimum down-time t_{down} of a unit. According to this constraint, a generator should operate t_{up} hours straight after becoming online, or it should stay offline t_{down} hours before starting to operate. When this constraint is violated by any generator, a penalty cost, named the up/down penalty, is added to the power generation cost. The up/down penalty constraint is formulated in Eq. 2.

$$\begin{array}{ll} if\ v_i(t) = 1 & x_i(t-1) \ge t_{down} \\ else & x_i(t-1) \ge t_{up} \end{array} \tag{2}$$

The fuel cost depends on the amount of power produced by each online generator for a given time slot. The predetermined power demand and the power reserve requirements need to be fulfilled for each time slot as given in Eq. 4 and Eq. 5 while keeping the generated power of each unit within its minimum and maximum values as in Eq. 6. For N generating units and T hours, the SEPGS objective function is shown in Eq. 3.

$$\min \quad F_{total} = \sum_{t=1}^{T} \sum_{i=1}^{N} \left[F_i(P_i(t)).v_i(t) + CS_i(t) \right] \tag{3}$$

subject to constraints in Eqs. 4-6:

$$\sum_{i=1}^{N} P_i(t).v_i(t) = PD(t) \tag{4}$$

$$\sum_{i=1}^{N} P_i^{\max}(t).v_i(t) \geq PD(t) + PR(t) \tag{5}$$

$$v_i(t).P_i^{\min} \leq P_i(t) \leq v_i(t)P_i^{\max} \tag{6}$$

The fuel cost of generating *p* MW power for the i-th unit is calculated using the Eq. 7. Fuel cost of a power unit *i* depends on three parameters, a_{0i}, a_{1i} and a_{2i}, which are predetermined for each generator. Commonly in literature, the lambda iteration technique [10] is used with this formulation to allocate the required load demand between the operating generators in each time slot while minimizing the total power generation costs.

$$F_i(p) = a_{0i} + a_{1i}.p + a_{2i}.p^2 \tag{7}$$

3 Application of Hyper-heuristics to the SEPGS

Many heuristics exist in literature to solve complex optimization problems. Since heuristics require information and experience about the problem, it is difficult to adapt a heuristic to a specific problem. Therefore, a single heuristic can not be developed which is applicable to a wide range of optimization problems. To overcome this limitation, hyper-heuristic methods [14] are introduced.

Selection hyper-heuristics differ from heuristics, since they do not directly operate on the solution space. Instead, they are used to select a heuristic from a set of low level heuristics, which will be applied on the solution space [15]. Hyper-heuristics do not need problem specific knowledge to operate; therefore, they are defined as problem independent methods. Therefore, they can be successfully applied to a wide range of optimization problems with varying data sizes [14].

A selection hyper-heuristic process contains two phases: heuristic selection and move acceptance [14, 15]. In the heuristic selection phase, a decision needs to be made for the heuristic, which will be applied to the current solution candidate, either randomly or with respect to certain performance indicators. The move acceptance mechanism decides whether the new candidate solution replaces the current solution according to the selected move acceptance criterion.

For the heuristic selection mechanism, different strategies are proposed [14, 15]. In this study, six different heuristic selection strategies are incorporated into the hyper-heuristic approach. These are: simple random (SR), random descent (RD), random permutation (RP), random permutation descent (RPD), greedy (GR) and choice function (CF).

Simple random heuristic selection strategy chooses a heuristic randomly. Random descent is similar, but the selected heuristic is applied to the solution repeatedly, until the solution cannot be improved anymore. In the random permutation strategy, a permutation array of heuristics is created randomly and the heuristics are applied to the solution in the provided order. In the random permutation descent, heuristics are applied repeatedly in the provided order, until they do not improve the solution. In some hyper-heuristic frameworks [15], the set of low level heuristics contains both mutational heuristics and hill climbers. If the selected heuristic is a mutational heuristic, then a hill climber is applied to the solution; otherwise, only a hill climber is applied and the fitness value of the resultant solution is calculated before the move acceptance phase.

Greedy methods apply all low level heuristics to the candidate solution at the same iteration and select the heuristic that creates the best solution [14, 15].

In Choice function heuristic selection method, a score is assigned to each heuristic. This score depends on three performance criteria [14, 15]. First criterion is the individual performance of a heuristic denoted with $f_1(h_i)$. Second criterion is the performance of a heuristic denoted with $f_2(h_i, h_k)$, when it is used in combination with other heuristics. The last criterion is the elapsed time since the last heuristic was used. At each iteration, these scores are calculated for each heuristic. The performances of the heuristics are computed as given in Eqs. 8-11.

$$f_1(h_i) = \sum_n \alpha^{n-1} (I_n(h_i) / T_n(h_i)) \tag{8}$$

$$f_2(h_i, h_k) = \sum_n \beta^{n-1} (I_n(h_i, h_k) / T_n(h_i, h_k)) \tag{9}$$

$$f_3(h_i) = \mathit{elapsedTime}(h_i) \tag{10}$$

$$\mathrm{score}(h_i) = \alpha * f_1(h_i) + \beta f_2(h_i, h_k) + \delta * f_3(h_i) \tag{11}$$

where $I_n(h_i,h_k)$ and $T_n(h_i,h_k)$ are the changes in the fitness function and the amount of time taken, respectively, when the n^{th} last time the heuristic h_k was applied after the heuristic h_i. In Eq. 11, α, β and δ are the relative weight factors used to calculate total scores of each heuristic.

For the move acceptance phase, four different strategies [14] are used in this paper. These are accept All Moves (AM), accept Only Improving Moves (OI), accept Improving and Equal Moves (IE) and the Great Deluge (GD) strategies. In the AM strategy, all moves are accepted. In the OI strategy, improving moves are accepted, but all non-improving moves are rejected. In the IE strategy, moves which improve the solution or does not change the fitness of the solution, are accepted.

In the GD move acceptance algorithm, the fitness of the initial solution is calculated and this value is set as the initial level value. Then, the down rate value is determined using Eq. 12. After a heuristic is applied to the candidate solution, the fitness value of the resultant solution is calculated. If the fitness value is better than the level value, the level is decremented by the DownRate value and the resultant solution is replaced with the current solution; otherwise, the resultant solution is discarded and the next heuristic is applied to the current solution.

$$\text{DownRate} = (f(s_0) - \text{BestResult}) / \text{Number of Iterations} \tag{12}$$

where BestResult is the best result found in literature for this problem so far and $f(s_0)$ is the fitness value of the initial solution.

4 Experiments

In this paper, selection hyper-heuristics with different heuristic selection and move acceptance strategy combinations are applied to the SEPGS problem. A binary representation is used with a length of N*T, where N is the number of units and T is the number of time slots. The variables in a solution take the values 0 or 1 to show that the corresponding power unit is either off or on for the corresponding time slot.

Seven operators are used as low level heuristics in the proposed approach. These operators are taken from [9], where they are used as mutation operators in a genetic algorithm applied to the same problem. These are: mutation with a probability of 1/L, mutation with a probability of 2/L, where L is the solution length, swap-window, window-mutation, swap-mutation, swap window hill-climbing, and Davis bit hill-climbing [16] operators. The last three operators combine mutational heuristics with hill climbers in a single heuristic.

4.1 Experimental Setup

The selection hyper-heuristics with twenty four heuristic selection and move acceptance combinations are tested on two benchmark problems taken from literature. These benchmarks are referred to as System 1 and System 2 in the rest of the paper. System 1, taken from [9, 10], consists of 10 units. System 2 data is generated by repeating the data of System 1 two times, as also done in [9, 10]. As a result, System 2 has 20 generators. For these two test systems, the time horizon is 24 hours.

For System 1 and System 2, the demand penalty and the up/down penalty coefficients are set to a very high value as 100000, so that infeasible solutions cannot have a better fitness than feasible ones. The number of allowed iterations for the hyper-heuristics per run is chosen as 1000 and 5000, for System 1 and System 2 respectively. These values are determined empirically. Best, average and worst results are reported for both systems over 20 runs. Furthermore, statistical tests are performed on the resultant solutions using the Matlab statistical tools for one-way ANOVA (analysis of variance) tests and for multiple comparison tests using the Tukey Honestly Significant Difference approach at a confidence level of 0.95.

4.2 Experimental Results

The best, average and worst cost values obtained by different combinations on two test systems are reported in Tables 2 through 9.

In the first experiment, OI move acceptance criterion is applied with six heuristic selection methods to System 1. RPD achieves the best result in this data set as shown in Table 2. RP comes second in front of SR and RD. In System 2, RPD again achieves the best result in front of the RP method. The difference percentage between RPD and RP is 0.021% in System 1, but this value is increased to 0.036% in System 2.

In the first two test instances, statistical tests show that there is a statistically significant difference between the fitness values obtained by different combinations and the mean values of GR and CF are significantly worse than the mean values of RPD, RP, SR and RD.

Table 2. Results for System 1 with OI

Algorithm	Best	Worst	Average
RPD	1125997	1128831	1127474
RP	1126231	1128931	1127689
SR	1127253	1129911	1128435
RD	1127253	1129563	1128572
CF	1127683	1148563	1133976
GR	1129038	1138217	1132815

Table 3. Results for System 2 with OI

Algorithm	Best	Worst	Average
RPD	2248284	2253971	2250434
RP	2249099	2253057	2250835
SR	2250116	2255899	2253378
RD	2250875	2253851	2252456
CF	2253743	2279271	2264473
GR	2255837	2267460	2258998

RPD and SR heuristic selection methods obtain the best result with the IE move acceptance criterion in System 1, but this result is not better than the best result produced by the RPD-OI strategy combination.

In System 2, RPD achieves the best result and RD obtains the second best result with IE. The difference between these results is very low, but they are not better than the best result obtained by the RPD-OI combination. These results show that the hyper-heuristic with OI move acceptance criterion produces more efficient results when compared with IE. According to the statistical test results, CF and GR have significantly worse mean values than the rest of the methods.

Table 4. Results for System 1 with IE

Algorithm	Best	Worst	Average
RPD	1126231	1129039	1127381
SR	1126231	1129317	1128190
RD	1127065	1130338	1128500
RP	1127253	1129837	1128510
CF	1128041	1147346	1135070
GR	1130520	1136545	1133359

Table 5. Results for System 2 with IE

Algorithm	Best	Worst	Average
RPD	2250070	2252741	2251331
RD	2250090	2254164	2252311
RP	2250837	2253019	2251510
SR	2250875	2253881	2251794
CF	2252492	2284777	2263464
GR	2255999	2263901	2260230

RPD also achieves the best result with GD in System 1. This result is equal to the best result obtained by the RPD-OI combination, but its average value is not better than the average value of RPD-OI. CF shows its best performance with GD and comes second in this data set.

In System 2, RPD, RD and SR obtain the best result with GD, but this result is not better than the result achieved by RPD-OI with System 2 data. RP follows these methods with its performance. Statistical results show that there is a statistically significant difference between the methods. Besides, RPD and CF have significantly different mean values when GD is used as the move acceptance criterion.

Table 6. Results for System 1 with GD

Algorithm	Best	Worst	Average
RPD	1125997	1129390	1127673
CF	1126119	1134568	1128820
RP	1126231	1129404	1127944
SR	1126231	1129837	1128267
RD	1127055	1129837	1128343
GR	1127252	1129135	1128345

Table 7. Results for System 2 with GD

Algorithm	Best	Worst	Average
RPD	2249099	2252103	2251066
RD	2249099	2253712	2251471
SR	2249099	2254148	2251906
RP	2249576	2253223	2251336
GR	2250904	2259784	2254414
CF	2251195	2272279	2259073

According to the results in Table 8 and Table 9, heuristic selection methods produce the poorest results with AM move acceptance criterion in both of the test systems. GR, which is the poorest heuristic selection method in the previous experiments, obtains the best results with AM, because it always selects the heuristic with the best resultant solution. Since hill climbers are also used as heuristics, they do not accept a worsening move, even if AM is used after the application of the hill climber. Therefore, a worsening move is never accepted by GR. The statistical test results do not indicate statistically significant differences.

Table 8. Results for System 1 with AM

Algorithm	Best	Worst	Average
GR	1135972	1182232	1157148
CF	1137093	1180722	1158591
RPD	1140067	1180180	1160381
RP	1141958	1180711	1161860
RD	1142190	1184874	1163611
SR	1152371	1183624	1165224

Table 9. Results for System 2 with AM

Algorithm	Best	Worst	Average
GR	2339024	2478087	2402021
CF	2341696	2482374	2404243
RPD	2348286	2477003	2406051
RD	2354096	2481037	2419543
RP	2356811	2483620	2418755
SR	2383415	2482194	2427434

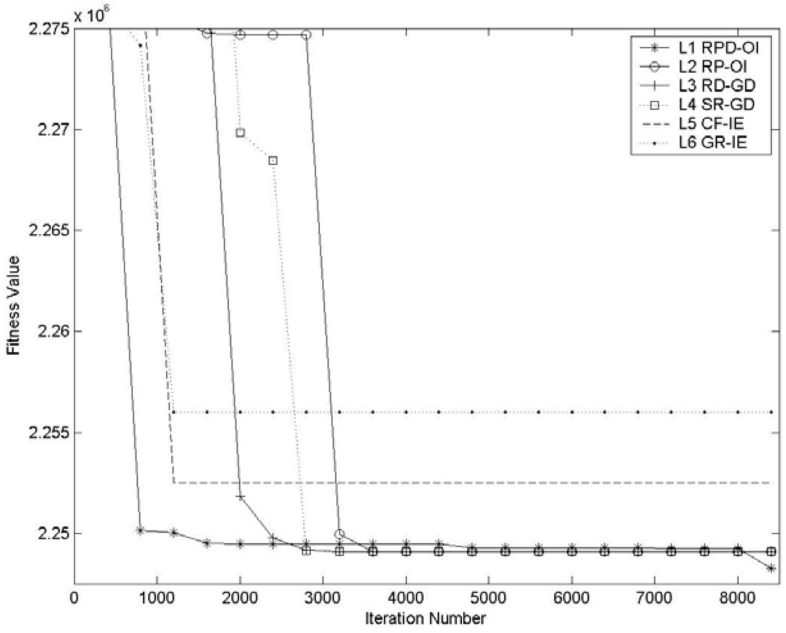


Fig. 1. Iteration Number vs. Fitness Value Curves

Commonly in literature, the GR and CF heuristic selection methods are among the better ones for most problems. However, in this problem, the best heuristic selection method is found to be RPD. We believe this is due to properties of the problem. To explore why this is so, we looked at the iteration number vs. the fitness value plots.

Fig. 1 illustrates the iteration number versus the fitness value curves for six different heuristic selection methods with respect to their best results in System 2. It can be seen that CF-IE and GR-IE achieve their best values much earlier than other combinations, but they get stuck at this local optimum, because these methods are greedy and they can not explore the whole search space efficiently. SR-GD, RD-GD and RP-OI achieve the second best result in System 2 and they find this result approximately between the iterations of 3000 and 3600. RPD-OI finds the overall best result approximately in the 8300th iteration. The randomness introduced by these methods helps the algorithm to be able to escape the local optima and to find better results later on in the search.

5 Conclusion

In this study, selection hyper-heuristics using six different heuristic selection methods and four move acceptance criteria are implemented and applied to the SEPGS problem. A set of experiments are executed on two data sets to find out the most effective strategy combination for this problem. In both of the problem instances, RPD-OI heuristic selection and move acceptance combination achieves the best results. Especially, its effectiveness becomes more emphasized in the second data set.

Aside from the observations mentioned above, it is also noticed that GR and CF methods obtain the poorest results in these experiments, although these two methods find their best values much earlier than the other heuristic selection methods. This shows us that using performance indicators about previous runs leads to a greedy method which is prone to get stuck at local optima and not able to explore different areas of the search space in an effective way.

Statistical tests show a statistically significant difference between the fitness values obtained by different heuristic selection methods for the same move acceptance criterion except for those using the AM acceptance scheme. Additionally, it also indicates that one heuristic selection method outperforms at least one of the other methods in the fitness values of the solutions it produces.

In our previous study [13], the results produced by RPD-OI are compared with the previously published results achieved by other optimization techniques. Because of the promising results obtained in these two studies, research will continue to enhance the hyper-heuristic approach. Advanced mutational heuristics and local search operators can be incorporated into the hyper-heuristic method to increase the effectiveness of this approach. Besides, the quality of the initial solution can be improved by using heuristics such as the priority list method [6]. Effective decision making techniques including a learning mechanism, which keeps information about the previous iterations, can be used in heuristic selection and move acceptance parts.

References

1. Lowery, P.G.: Generating Unit Commitment by Dynamic Programming. IEEE Transactions on Power Apparatus and Systems PAS-85(5), 422–426 (1966)
2. Cheng, C., Liu, C.W., Liu, C.C.: Unit Commitment by Lagrangian Relaxation and Genetic Algorithms. IEEE Transactions on Power Systems 15(2), 707–714 (2000)
3. Mantawy, A.H., Abdel-Magid, Y.L., Selim, S.Z.: Unit Commitment by Tabu Search. IEEE Proceedings – Generation, Transmission and Distribution 145(1), 56–64 (1998)
4. Zhuang, F., Galiana, F.D.: Unit Commitment by Simulated Annealing. IEEE Transactions on Power Systems 5(1), 311–318 (1990)
5. Chen, C.L., Wang, S.C.: Branch-and-Bound Scheduling for Thermal Generating Units. IEEE Transactions on Energy Conversion 8(2), 184–189 (1993)
6. Burns, R.M., Gibson, C.A.: Optimization of Priority Lists for a Unit Commitment Program. In: Proceedings of IEEE/PES Summer Meeting, pp. 1873–1879 (1975)
7. Viana, A., de Sousa, J.P., Matos, M.: Using Grasp to Solve the Unit Commitment Problem. Annals of Operations Research 120, 117–132 (2003)
8. Simon, S.P., Padhy, N.P., Anand, R.S.: An Ant Colony System Approach for Unit Commitment Problem. Electrical Power and Energy Systems 28, 315–323 (2006)
9. Kazarlis, S.A., Bakirtzis, A.G., Petridis, V.: A Genetic Algorithm Solution to the Unit Commitment Problem. IEEE Transactions on Power Systems 11(1), 83–92 (1996)
10. Valenzula, J., Smith, A.E.: A Seeded Memetic Algorithm for Large Unit Commitment Problems. Journal of Heuristics 8, 173–195 (2002)
11. Maturana, J., Riff, M.C.: Solving the Short-term Electrical Generation Scheduling Problem by an Adaptive Evolutionary Approach. European Journal of Operational Research 179, 677–691 (2007)
12. Uyar, Ş.A., Türkay, B.: Evolutionary Algorithms for the Unit Commitment Problem. Turkish Journal of Electrical Engineering 16(3), 239–255 (2008)
13. Berberoğlu, A., Uyar, Ş.A.: A Hyper-Heuristic Approach for the Unit Commitment Problem. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Ebner, M., Farooq, M., Fink, A., Grahl, J., Greenfield, G., Machado, P., O'Neill, M., Tarantino, E., Urquhart, N. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 121–130. Springer, Heidelberg (2010)
14. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: A Survey of Hyper-heuristics. Computer Science Technical Report, University of Nottingham, NOTTCS-TR-SUB-0906241418-2747 (2009)
15. Ozcan, E., Bilgin, B., Korkmaz, E.E.: A Comprehensive Analysis of Hyper-heuristics. Intelligent Data Analysis 12(1), 3–23 (2008)
16. Davis, L.: Bit Climbing Representational Bias and Test Suite Design. In: Proceedings of the 4th International Conference on Genetic Algorithms, pp. 18–23 (1991)

A Genetic Algorithm for Radiotherapy Pre-treatment Scheduling

Sanja Petrovic and Elkin Castro

Automated Scheduling Optimisation and Planning (ASAP) Research Group,
School of Computer Science, University of Nottingham,
Nottingham, NG8 1BB, UK
{sxp,edc}@cs.nott.ac.uk

Abstract. This paper is concerned with the radiotherapy pre-treatment patient scheduling. Radiotherapy pre-treatment deals with localisation of the area to be irradiated and generation of a treatment plan for a patient. A genetic algorithm is developed for patient scheduling which evolves priority rules for operations of radiotherapy pre-treatment. The fitness function takes into consideration the waiting time targets of patients and also the early idle time on resources. Real world data from a hospital in the UK are used in experiments.

Keywords: Patient scheduling, radiotherapy, multiple objectives, evolutionary algorithm.

1 Introduction

Radiotherapy is often used as means to treat cancer patients. Radiotherapy includes two phases: pre-treatment, and treatment on linac (linear particle accelerator) machines which deliver radiation. This research focuses on the radiotherapy pre-treatment scheduling of patients. The purpose of the pre-treatment is to define the precise area to be treated with radiotherapy, and to generate a radiotherapy treatment plan which targets the tumour while keeping the radiation to the surrounding healthy tissues and organs to a minimum. In radiotherapy, radiation is given as a series of small doses, referred to as fractions, over a period of days or weeks, although it can also be given as a single fraction.

The consequences of long waiting times for radiotherapy on patients have been discussed in the literature and medical community [5,8,10]. Long waiting times for radiotherapy treatment unfavourably affect both the possibility of cure and the reduction of the severity symptoms by allowing tumour growth and causing anxiety. It has been recommended to expand the resources (staff and machines) to meet the growing demand for radiotherapy services. However, these resources are expensive and it takes time to have them ready [5,16]. Therefore, the objective is to find ways to improve the use of existing resources while keeping the quality of treatment at satisfactory levels by using better scheduling procedures.

Radiotherapy scheduling of patients on linac machines has attracted research interests in the last decade or so. Approaches of different nature have been

© Springer-Verlag Berlin Heidelberg 2011

developed, starting from mathematical programming models [4], to heuristic approaches [14] and constructive approaches hybridised with GRASP (Greedy Random Adaptive Search Procedure) [13].

The pre-treatment and radiotherapy scheduling problems were considered together. In [9,15] the authors formulate a simulation model. Their results showed that performance of the schedule could improve subject to having more resources or working extra time. In [12] the authors described a genetic algorithm where schedules, for both phases considered together, were encoded using an operation-based representation. This research work was focused mostly on the investigation of availability of doctors to approve the treatment plans.

In our earlier research work, we investigated the hybridisation of integer linear programming and priority rules where a daily scheduling approach is considered. A schedule generated by priority rules was used as an initial solution for the CPLEX model. Although CPLEX improved the initial solutions provided by priority rules, the overall performance of schedules, over a period of 120 weeks, was worse than that of the schedules generated by priority rules only. In the research presented in this paper, we aim to balance the effect of the greedy optimisation method whose intent is to utilise all the capacity of resources, by studying the effect of penalising early idle time on resources. This approach was firstly developed for dynamic optimisation in [3].

A genetic algorithm is developed which considers the idle time on resources within a predefined time window in the fitness function. This algorithm evolves priority rules which are used to decode chromosomes into schedules using the Giffler and Thompson's algorithm presented in [17]. Real world data provided by the Nottingham University Hospitals NHS Trust, City Hospital Campus are used to evaluate this method.

This paper is organised as follows. Section 2 introduces the radiotherapy pre-treatment scheduling problem present at the City Hospital Campus, Nottingham. Section 3 explains the implemented genetic algorithm. Section 4 introduces the problem instances used in the experiments and discusses the results. Finally, Section 5 gives conclusions and indicates future research directions.

2 Problem Statement

Radiotherapy patients are classified according to a treatment intent and a waiting list status, and are diagnosed with a site (area to be treated). The treatment intent can be palliative or radical. A palliative treatment is meant to relieve symptoms, while the intent of a radical treatment is to cure. There are three waiting list statuses: emergency, urgent and routine. The waiting list status is determined by the site and progress of the cancer. The waiting time target determines the date by which a patient needs to have their first radiotherapy fraction (i.e. the date by which pre-treatment has to finish). Table 1 shows the Joint Council for Clinical Oncology (JCCO) good practice and maximum acceptable waiting time targets given in days (separated by "/") [8]. On the other hand, the UK Department of Health (DH) suggests a 31-day waiting time target for all cancers [5].

Table 1. JCCO recommended waiting time targets

Intent	**Waiting list status**		
	Emergency	**Urgent**	**Routine**
Palliative	1/2	2/14	2/14
Radical	–	–	14/28

Table 2. Radical head and neck pre-treatment pathway

Operation	**Resources**	**Description**	**Processing time**	**Lead time (days)**
1	Mould room	Beam directed shell (BDS)	60 min	1
2	CT scanner	CT scan with BDS	45 min	0
3	Doctor	Planning	60 min	0
4	Physics unit	Isodose plan	2 days	0
5	Doctor and simulator	Treatment verification	30 min	0
6	Doctor	Approve verification and prescribe dose	20 min	0
7	Physics unit	Plan check	30 min	0
8	Verification system	Enter data	30 min	0
9	Verification system	Check data	30 min	0

The site of the cancer and treatment intent determine the patient pre-treatment pathway. The City Hospital treats a large number of cancer sites including head and neck, breast, prostate, etc. Table 2 shows as an example the pathway of a radical head and neck patient. First, the patient visits the mould room, where a bean directed shell (BDS) is fit. The BDS takes one hour to make, but it will only be ready one day after it has been moulded. This extra time is referred to as lead time. When the BDS is ready, the patient visits the CT scanner which is used to decide the precise area to be treated. Once the scan is available the doctor can generate a treatment plan together with the physics unit. This procedure takes two working days to complete as it needs discussion between the doctor and the physics unit. However, the doctor and the physics unit are available for other operations. After the treatment plan is complete, the patient goes to the simulator where the radiotherapy treatment position and plan are checked. Depending on the case, the doctor has to be present during the treatment verification. After the treatment verification, the doctor approves the verification and prescribes the radiation dose. Finally, the radiotherapy treatment is checked by the physics unit, before the corresponding data are entered and checked onto the verification system. We note that a single doctor is assigned to a given patient, and that this doctor must be a specialist for the site of the patient. The exception to this rule are palliative patients who can be treated by any doctor.

Resources have limited availability. All resources, except doctors, are continuously available throughout the clinic opening hours: Monday to Friday from 8:30 to 18:00, and weekends from 9:00 to 13:00. Doctors availability is defined from Monday to Friday for specified operations and patient types. During weekends doctors are available on call.

Palliative-emergency and palliative-urgent patients can be seen on any day. On the other hand, radical and palliative-routine patients can be treated only from Monday to Friday.

Table 2 shows that this problem has resource concurrency (an operation may require two or more resources simultaneously) and recirculation (a given resource is used more than once in the same pathway).

3 Genetic Algorithm

A genetic algorithm is developed whose main characteristics are:

- The fitness function includes a term that penalises early idle time within a predefined time window [3].
- Schedules are decoded using the modified Giffler and Thompson's algorithm given in [17]. In this algorithm, parameter δ modulates the size of the search space. If $\delta = 1$ the algorithm produces active schedules. If $\delta = 0$ the algorithm generates non-delay schedules. For values of δ from (0, 1) schedules from a subset of active schedules, including all non-delay ones, are generated. In the proposed genetic algorithm δ is not fixed as for example is the case in [1,3,11] but this parameter is evolved as in [7].

Fitness Function. We assume that on a given day there is a set $\mathcal{P}$ of patients to be scheduled. The quality of the schedule is evaluated by a fitness function which has two terms. The first term considers waiting time targets of the patients while the second term takes into account the idle time on the resources. The first term consists of three objectives. The first one is the weighted number of patients exceeding the waiting time targets (1). Priority weights of patients w_j depend on the patients' waiting list status. Binary variable U_j is equal to 1 if patient P_j exceeds the waiting time target, and it is 0 otherwise. The second objective is the maximum lateness (2). The lateness of a patient is calculated as the difference between the waiting time target and the completion date of the pre-radiotherapy treatment. The third objective is the sum of weighted lateness (3).

$$\text{Minimise} \quad z_1 = \sum_{P_j \in \mathcal{P}} w_j \, U_j \ . \tag{1}$$

$$\text{Minimise} \quad z_2 = \max_{P_j \in \mathcal{P}} \{L_j\} \ . \tag{2}$$

$$\text{Minimise} \quad z_3 = \sum_{P_j \in \mathcal{P}} w_j \, L_j \ . \tag{3}$$

Each objective (1)-(3) is normalised to take values in the $[0,\ 1]$ interval with respect to the largest and smallest values of that objective in a given population, according to (4). In (4) v^k is the value of chromosome k to be normalised, $\overline{v}^k$ is its normalised value, and l is the index for chromosomes in the population.

$$\overline{v}^k = \frac{v^k - \min_l\{v^l\}}{\max_l\{v^l\} - \min_l\{v^l\}} \ . \tag{4}$$

The normalised values of (1)-(3) are used in the first term of the fitness function.

$$z^k = W_1\overline{z}_1^k + W_2\overline{z}_2^k + W_3\overline{z}_3^k \ . \tag{5}$$

Weights W_i $(i = 1,\ 2,\ 3)$ represent the relative importance of objectives (1)-(3). The value given by (5) is normalised to take values in the $[0,\ 1]$ interval using formula (4) before being included in the fitness function.

The second term of the fitness function corresponds to the early idle time on the resources. The idle time is calculated within a predefined time interval, denoted by ω, that starts on the day patients arrive for scheduling. The idle time is normalised to take values from the $[0,\ 1]$ interval according to (4). Let $\overline{\tau}^k$ be the normalised resource idle time of chromosome k and $\alpha \in [0,\ 1]$ its importance in the fitness function, then the fitness function of chromosome k is given as.

$$f^k = \overline{z}^k + \alpha\,\overline{\tau}^k \tag{6}$$

Parameters α and ω are used to control the balance between the performance of the schedule in terms of the utilisation of resources and having some spare capacity for future patients.

Encoding and Decoding. A schedule is encoded by using the priority rule-based representation given in [6]. In our implementation, the length of the chromosome is equal to the total number of operations. We also include one additional gene that stores $\delta \in [0,\ 1]$.

Chromosome k is represented by string $(\pi_1^k,\ \pi_2^k, \ldots,\ \pi_M^k,\ \delta^k)$ where M is the total number of operations and π_t is a rule from the set of predefined priority rules.

Chromosomes are decoded into solutions using the modified Giffler and Thompson's algorithm introduced in [17]. Let $\mathcal{C}_t$ be the set of operations ready to be scheduled in iteration t. Operation $O_t \in \mathcal{C}_t$ with earliest completion time on the required resource is selected; ties are broken arbitrarily. A conflict set $\mathcal{G}_t$ is defined with all operations from $\mathcal{C}_t$ which require O_t's resource and whose processing time overlaps with that of O_t. A smaller (respectively larger) value of δ means a smaller (respectively larger) size of the conflict set $\mathcal{G}_t$. Priority rule π_t is used to select an operation from $\mathcal{G}_t$ to be scheduled next. The pseudo-code is given in Algorithm 1.

In order to illustrate the availability of resources calculated in step 5, let us assume as an example that O_t is treatment verification (see Table 2). Then, m_t will be the doctor, as this resource has less hours of availability for treatment verification per week than the simulator.

Algorithm 1. Modified Giffler and Thompson's algorithm [17]

```
1: $t \leftarrow 1$
2: $\mathcal{C}_t$ is the set of operations ready to be scheduled
3: while $\mathcal{C}_t \neq \emptyset$ do
4:     $O_t \leftarrow$ operation from $\mathcal{C}_t$ with earliest completion time $\phi_t$
5:     $m_t \leftarrow$ a resource required by $O_t$. If $O_t$ requires multiple resources, $m_t$ is the least
       available resource
6:     $\sigma_t \leftarrow$ earliest start time on machine $m_t$
7:     $\mathcal{G}_t \leftarrow \{O \in I(m_t) : \sigma(O) \leq \sigma_t + \delta(\phi_t - \sigma_t)\}$, where $I(m_t)$ is the set of operations
       ready to be scheduled on $m_t$ and $\sigma(O)$ is the earliest start time of operation $O$
8:     Choose operation $O^* \in \mathcal{G}_t$ by using priority rule $\pi_t$
9:     Schedule $O^*$
10:    Update $\mathcal{C}_t$
11:    $t \leftarrow t + 1$
12: end while
```

Variation Operators. Figure 1 displays the implemented one point crossover. Genes δ are recombined by means of two convex combinations, one for each offspring. The recombined values of δ are: $\delta'_1 = \delta_1\lambda + \delta_2(1-\lambda)$ and $\delta'_2 = \delta_1(1-\lambda) + \delta_2\lambda$. Value λ is chosen randomly from the $[0,\ 1]$ interval. A convex combination is chosen as a crossover rule because its definition guarantees that $\delta'_1,\ \delta'_2 \in [0,\ 1]$.

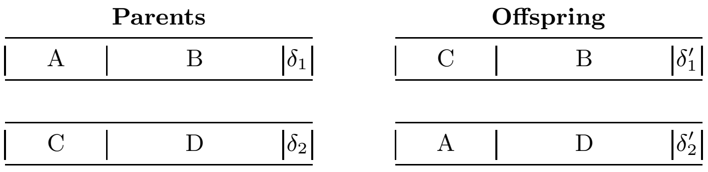


Fig. 1. Implemented one point crossover

A chromosome is mutated by randomly selecting one of its genes. If this gene is a priority rule, it is replaced by a randomly selected rule from the set of predefined rules. If the selected gene is δ, a new value is randomly chosen from the $[0,\ 1]$ interval.

4 Experimental Results

The City Hospital provided us with two sets of data on patients. The first data set contains five years of data with the arrival date, treatment intent and waiting list status of patients. A second set has data from 188 patients which contains all the details about the patients including the site and doctor booked for each patient. These two data sets are combined to give different problem instances.

The values of the genetic algorithm parameters are as follows. The crossover probability is 0.6, the mutation probability is 0.1, and the population size is

100 chromosomes [1,3,11]. The genetic algorithm is allowed to run after a given number of generations has been reached with no improvement in the fitness function. This number is equal to the number of patients in the given daily scheduling problem [1]. The fitness proportionate selection with an elitist strategy of one individual is used. Chromosomes of the initial population are randomly created, namely each gene representing a priority rule is initialised by randomly selecting a rule from a set of 44 rules, while δ is randomly selected from a $[0,\ 1]$ interval.

We consider a comprehensive set of priority rules which can be classified as in [2]. Rules that involve waiting time targets such as: earliest waiting time target, slack-based rules, etc; rules involving the processing time: shortest processing time, least work remaining, fewest remaining operations, etc; rules that depend on characteristics other than processing time and waiting time targets such as: random selection and arrived at queue first, and rules that depend on two or more characteristics such as: slack per number of operations remaining, slack per work remaining, etc.

Priority weights w_j of emergency, urgent and routine patients are 5, 2, and 1 respectively. Weights W_i of the objectives (1), (2) and (3) are 6, 3, and 1. These values are set in consultation with the Hospital staff, but are subjective.

The aim of the experiments is to investigate the effect of the parameters α (weight of the early idle time in the fitness function) and ω (the time window within which early idle time is penalised) on the performance of the schedule. Following the practice of oncology departments in the UK the performance of the schedule is measured as the percentage of the patients who breach the JCCO and the DH waiting time targets.

The system is warmed up for two years (i.e. the system is loaded with patients during two years but the generated schedule is not used in the system evaluation). Starting with the generated booking system, the performance of the schedule is measured every four weeks, because the Hospital reports performance on a monthly basis, in the following 136 weeks (it gives us 34 experiments, each one is four weeks long). The average values of the percentage of patients who breach the waiting time targets over 34 experiments are reported.

Table 3 displays the average percent of patients who breach the JCCO maximum acceptable waiting time targets and DH waiting time targets for combinations of α and ω.

For both the JCCO and the DH waiting times targets we can notice that, larger improvements happen for values of α below 0.5 and large values of ω. In particular, the largest improvements are achieved at certain combinations of $\alpha \in \{0.2,\ 0.3,\ 0.4\}$ and $\omega \in \{4,\ 7\}$. On the other hand, by having large values of both α and ω, the early idle time is given higher relative importance with respect to the first term that considers waiting time targets in the fitness function. This bias prevents the algorithm from reaching better average performance over the time period of 136 weeks. The best achieved results are underlined and boldfaced, while the "good" ones are boldfaced only.

The gained improvements are very sensitive to changes in the the values of α and ω. For example, pair $(\alpha,\ \omega) = (0.3,\ 7)$, which produces the lowest percent

Table 3. Average percent of patients who breach the waiting time targets over 34 observations

	JCCO			DH		
α	$\omega = 1$	$\omega = 4$	$\omega = 7$	$\omega = 1$	$\omega = 4$	$\omega = 7$
0.0	—	8.66	—	—	6.10	—
0.1	8.61	8.66	8.66	6.10	6.10	6.10
0.2	8.42	**8.33**	8.95	**5.87**	**5.74**	6.44
0.3	8.65	8.62	**<u>8.25</u>**	6.27	6.06	**<u>5.74</u>**
0.4	**8.39**	**8.34**	**8.39**	**5.87**	**5.74**	**5.95**
0.5	8.63	8.46	8.63	6.10	5.96	6.10
0.6	8.77	8.40	8.66	6.27	5.96	6.10
0.7	**8.37**	8.59	8.63	**5.87**	6.06	6.10
0.8	8.71	8.43	8.83	6.10	5.96	6.27
0.9	8.62	8.65	8.68	6.06	6.06	6.10
1.0	**8.37**	8.63	8.62	**5.87**	6.10	6.06

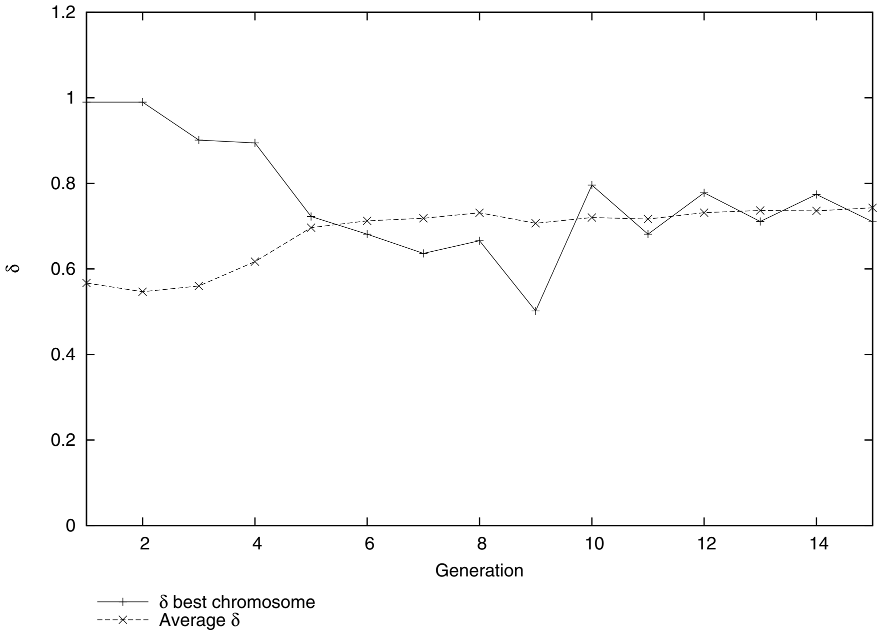


Fig. 2. Evolution of δ

values, is in the neighbourhood of combinations which the largest percent breaches. In particular, slightly decreasing the value of α, in a combination where α is small and ω is large, leads to a increase in the percent of patients breaching the waiting time targets. Further analyses need to be carried out to understand this undesirable sensitivity.

Regarding δ, as an example, Fig. 2 shows the value of δ of the best chromosome and its average value across the population, for a given scheduling problem. The algorithm stopped after 15 generations. We can see that the quality of the schedules benefits by modulating the size of the search space by limiting the size of the conflict set $\mathcal{G}_t$. Namely, in the initial population the best chromosome has $\delta = 0.989858$ while at the end it has the value $\delta = 0.710254$.

5 Conclusion and Future Work

We presented a genetic algorithm for a radiotherapy pre-treatment scheduling problem. The fitness function is a combination of two terms. One term considers the waiting time targets of patients and the other penalises the early idle time within a defined number of days. We investigated the effect of the weight of the early idle time in the fitness function and the length of time during which early idle time is penalised on the performance of the schedules. Higher improvements are achieved with values of α under 0.5 and large values of ω. However, settings in α and ω are sensitive to small changes in α.

The system has been demonstrated to the hospital and will be left for further evaluation. Our future work will be focused on understanding the sensitivity to changes in α and ω. Additionally, other representations should be studied. Priority rule-based encoding may lead to false competition (this happens when multiple individuals produce the same schedule). Alternatively, we will investigate a preference list-based encoding, where each resource has its own priority list of patients.

Acknowledgements. The authors would like to thank the Engineering and Physics Science Research Council (EPSRC), UK, for supporting this research (Ref. No. EP/NB2076949511/1). We would also like to acknowledge the support of the Nottingham University Hospitals NHS Trust, City Hospital Campus, UK.

References

1. Bierwirth, C., Mattfeld, D.: Production Scheduling and Rescheduling with Genetic Algorithms. Evolutionary Computation 7(1), 1–17 (1999)
2. Blackstone Jr., J., Phillips, D., Hogg, G.: A state-of-the-art survey of dispatching rules for manufacturing job shop operations. International Journal of Production Research 20(1), 27–45 (1982)
3. Branke, J., Mattfeld, D.: Anticipation in Dynamic Optimization: The Scheduling Case. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., Schwefel, H. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 253–262. Springer, Heidelberg (2000)
4. Conforti, D., Guerriero, F., Guido, R., Veltri, M.: An optimal decision-making approach for the management of radiotherapy patients. OR Spectrum 33(1), 123–148 (2011)
5. Department of Health: The NHS Cancer Plan: a plan for investment, a plan for reform (2000)

6. Dorndorf, U., Pesch, E.: Evolution based learning in a job shop scheduling environment. Computers & Operations Research 22(1), 25–40 (1995)
7. John, D.: Co-evolution With The Bierwirth-Mattfeld Hybrid Scheduler. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), p. 259. Morgan Kaufmann Publishers Inc., San Francisco (2002)
8. Joint Council for Clinical Oncology: Reducing Delays in Cancer Treatment: Some Targets (1993)
9. Kapamara, T., Sheibani, K., Petrovic, D., Hass, O., Reeves, C.: A simulation of a radiotherapy treatment system: A case study of a local cancer centre. In: Proceedings of the ORP3 Conference, pp. 29–35 (2007)
10. Mackillop, W.: Killing time: The consequences of delays in radiotherapy. Radiotherapy and Oncology 84(1), 1–4 (2007)
11. Mattfeld, D., Bierwirth, C.: An efficient genetic algorithm for job shop scheduling with tardiness objectives. European Journal Of Operational Research 155(3), 616–630 (2004)
12. Petrovic, D., Morshed, M., Petrovic, S.: Genetic Algorithm Based Scheduling of Radiotherapy Treatments for Cancer Patients. In: Combi, C., Shahar, Y., Abu-Hanna, A. (eds.) AIME 2009. LNCS, vol. 5651, pp. 101–105. Springer, Heidelberg (2009)
13. Petrovic, S., Leite-Rocha, P.: Constructive and GRASP Approaches to Radiotherapy Scheduling. In: Ao, S. (ed.) Advances in Electrical and Electronics Engineering (IAENG) Special Edition of the World Congress on Engineering and Computer Science 2008 (WCECS), pp. 192–200. IEEE Computer Society, Los Alamitos (2008)
14. Petrovic, S., Leung, W., Song, X., Sundar, S.: Algorithms for radiotherapy treatment booking. In: Qu, R. (ed.) Proceedings of The Workshop of the UK Planning and Scheduling Special Interest Group, PlanSIG (2006)
15. Proctor, S., Lehaney, B., Reeves, C., Khan, Z.: Modelling Patient Flow in a Radiotherapy Department. OR Insight 20(3), 6–14 (2007)
16. Robinson, M.: Radiotherapy: technical aspects. Medicine 36(1), 9–14 (2008)
17. Storer, R., Wu, S., Vaccari, R.: New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling. Management Science 38(10), 1495–1509 (1992)

Planning and Optimising Organisational Travel Plans Using an Evolutionary Algorithm

Neil Urquhart

Centre for Emergent Computing, Edinburgh Napier University, Edinburgh, UK
n.urquart@napier.ac.uk

Abstract. Commuting to the workplace is a highly individualistic experience, especially where the private car is the chosen mode of transport. The costs of using cars with low occupancy rates are significant in environmental terms as well as requiring the provision of parking space at the workplace. This paper examines the use of an Evolutionary Algorithm based problem solver to construct travel plans for three sites with 248,404 and 520 employees respectively at each site. Results presented suggest that a significant saving in overall distance travelled and parking spaces required is possible. The algorithm employed takes into account both hard constraints and soft constraints (such as work patterns and journey flexibility).

1 Introduction and Motivation

Commuting to the workplace by private car is a major contributing factor to motoring related greenhouse gas emissions and to rush hour congestion in towns and cities. The provision of parking spaces can be a major cost to organisations who are based in areas with high land values. By having employees travel to work in groups, sharing a car, the pollution and congestion may be reduced and the number of parking spaces at the workplace also reduced. Planning car sharing requires the identification of groups of employees who live at set of addresses that can be served by one commute that is shorter than the collective individual journeys.

A major drawback to the automated planning of car sharing schemes is their inability to take into account the personal preferences of the individuals participating. The author previously undertook a study into such 'soft factors' in [2]. Some preliminary work on the concept of using an Evolutionary Algorithm to optimise car sharing was presented by the author in [8], this study presents a more in depth discussion of the algorithm and the results obtained. The algorithm used in this study minimises the toal distance travelled by cars, minimises the extent to which individuals deviate from their direct route and ensures that individuals share have similar work patterns.

2 Previous Work

There are many Vehicle Routing Problem (VRP) variants which have been identified and investigated by researchers. An overview of vehicle routing problems and the range of heuristics applied to them may be found in [7]. Disregarding soft preferences, the

© Springer-Verlag Berlin Heidelberg 2011

problem under discussion may be formulated as a Capacitated Vehicle Routing Problem with Time Windows (CVRPTW). The principle difference being vehicles start from a "customer" and are routed to the central workplace, rather than starting and ending at the same central location. A notable recent approach to the Vehicle Routing Problem with Time Windows (VRPTW) is presented in [4], the problem is formulated as a multi-objective problem, and the EA employed uses ranking to evaluate the population. This work is further developed in [6] where it is applied to the problem of routing garbage collections. The ranking approach has the potential to produce a range of solutions, which collectively form a Pareto front. In this case a strategy is required to determine which solution should be adopted by the user.

There exists several approaches to the problem of car sharing. Naor [3] formulates the problem around groups of employees using their own cars to drive to a intermediate meeting point, and continuing to the place of work in one vehicle. This approach is significantly different to the problem being discussed in this document. Naor examines the possibilities of optimising the sharing of driving equally. The 2nd leg optimised such that from the entire pool each driver does a fair share of 2nd leg driving. Buchholz [1] presents a car-pool problem as an NP complete partitioning problem. The system formulates detours to individuals' journeys to allow them to pick up other individuals. No results are presented, not are the preferences of individual users taken into account.

A games-theory approach to public transport planning and integration is explored in [5]. The authors take a market based approach using games theory to establish how public transport artefacts such as bus services should be provided in order to secure the maximum benefit from them.

3 Problem Description

The problem under consideration here is the construction of a potential car-sharing plan for a UK-based University. Employee address data from the University payroll system may be combined with UK-based Ordnance Survey geo-spatial data (obtained for academic use under the Digimap Agreement) to allow distances between employees homes and work place to be estimated. The aim of any solution is to group employees into groups of up to 4 individuals, who may share the same car. The car is provided by the employee living furthest away who deviates from their journey to pick up other members of the group. In this case the system allows users to specify a work pattern constraint (start and end times of their working day) and degree to which they will deviate from their direct journey in order to share.

The prototype optimises with respect to three objectives:

- The total distance travelled by cars should be minimised.
- The additional distance travelled by any employee should not exceed the deviation constraint specified by that employee.
- Each employee may specify a time slot that they work in, employees should only share with others in that time slot.

4 The Planning Algorithm

The test bed system constructed utilises an Evolutionary Algorithm (EA) to produce a solution in the form of a travel plan. The solution must divide employees into groups of up to four, each group sharing one car. The solution must take into account the objectives outlined (see section 3). There may exist no ideal solution, but rather the challenge is to find a compromise solution that satisfies as many of the objectives as possible.

The EA uses an indirect representation, each individual is not itself a full solution, but a list of groups, each employee being allocated to one group, the groups are not ordered internally. The number of groups may differ between individuals (but the total number of employees will always remain the same) as solutions may vary the number of passengers in each car. The minimum number of groups being determined by $\frac{noOfEmployees}{4}$), the maximum being $noOfEmployees$ (the case where each individual travels to work in a separate vehicle). The ordering of the groups within the chromosome is not significant. The role of EA is to form such groupings and preserve useful groups through the generational cycle.

The algorithm used maintains a steady-state population of 25 individuals, during each generation a sub-population of 15 children is created. A child may be created from two parents, via recombination, or by cloning a single parent. The probability of recombination is 0.8, parents are selected using a tournament of size 2. Each child has a mutation applied to it, one of the three operators outlined above. The fitness of the child is calculated by building a solution from the encoding within that child as outlined below. Members of the child population are copied into the main population, replacing the looser of a tournament.

Recombination consists of creating a new individual by adding groups selected alternately from each parent. As each group is added employees are removed from the group if they have been added previously as part of a group from the other parent. This encourages the retention of group membership across generations.

The following three mutation functions are employed to modify groups within the selected individual:

- Swap two employees between two groups.
- Create a new group. An existing group is selected (must have a minimum membership of 2 persons) and half its members are moved to the new group.
- Two groups are selected at random and merged (the combined membership of the groups must be equal to or less than 4 persons).

In order to evaluate the fitness of an individual a complete solution must be constructed an evaluated. A solution is constructed by applying an ordering heuristic to each employee group, to order them by distance from the workplace.

A penalty fitness value is then calculated for each group based on the following:

$$gFit = d + (d * devP) + (d * tpP - 1)$$

where:

gFit = the fitness value for the current group of employees
d = total distance driven by the group in meters
devP = 1 if any of the individuals in the group have a journey length that violates their deviation constraint
tpP = the number of different timeSlots within the group

The total fitness value for a candidate solution is the sum of the group fitness values, the more constraints that are broken within a solution the higher the fitness allocated to it. The algorithm is executed until 100 generations have elapsed without any fitness improvement.

Table 1. The datasets used in this investigation

Location	Employees	Average direct distance to workplace(km)
Site 3	404	6.2
Site 2	248	6.5
Site 1	520	6.0

Table 2. Results obtained whilst altering the deviation constraint

	Deviation (%)	Site 3	Site 2	Site 1
% Dist Saved	10	39.9	42.3	35.1
	30	44.9	45.3	39.7
	50	47.1	48.3	41.3
	70	48.7	48.5	43
	90	48.6	50.9	42.7
Parking spaces	10	44.9	39.4	53.3
	30	36.3	34.1	43.5
	50	32.7	31.8	39.5
	70	30.6	32.1	36.1
	90	30.61	29.7	35
Deviation Constraint Violations	10	4.8	4.9	4.3
	30	0.8	0	1.4
	50	0.5	0	0.6
	70	0	0.4	0.4
	90	0	0	0.1
Time slot Constraint Violations	10	0.4	0	0.6
	30	0.3	0	1.0
	50	0.3	0.1	0.8
	70	0.3	0.1	0.9
	90	0.3	0	1.2
Average Car Occupancy	10	2.2	2.5	1.9
	30	2.8	2.9	2.3
	50	3.1	3.1	2.5
	70	3.3	3.1	2.8
	90	3.3	3.4	2.9

5 Experimental Method and Results

The test bed system has been tested using data based on the payroll of a UK based University. Three university campuses were examined; the numbers employed at each site may be seen in table 1.

For testing purposes each individual is allocated a random work pattern identifier in the range (1..4) to represent their work pattern and a random deviation value in the range (10-90%).

The reader should consider that this problem may be presented as a ”design” type problem, where sufficient time is available to allow multiple runs to be made in order to take account of the stochastic nature of the algorithm. However in many cases it may be necessary to produce travel plans quickly (in order to respond to users’ changing requirements) in which case there may only be time in which to build one solution. With this in mind, the results presented here represent the average over 10 runs.

Over the following series of runs the deviation constraint was altered through values of 10,30,40, 70 and 90% for each user. For instance if a user has a direct journey distance of 10 kilometres to their place of work, and a deviation constraint of 30% then the maximum acceptable journey distance to them when participating in the car sharing scheme would be 13 kilometres. The results obtained may be seen in table 2. Note how as the deviation constraint is relaxed the other constraints are met, this ability to ”trade off” conflicting constraints is a well known feature of evolutionary algorithms.

Table 3. Results obtained whilst altering the number of work pattern

	Work Patterns	Site 3	Site 2	Site 1
Distance saved	1	64.9	65.1	64.0
	2	58.1	59.7	54.6
	3	55.7	55.4	49.7
	4	51.6	52.5	46.5
Parking spaces	1	25.4	25.5	25.4
	2	26.3	26.5	26.4
	3	27.0	27.3	27.0
	4	27.6	28.2	30.0
Deviation Constraint Violations	1	0	0	0
	2	0	0	0
	3	0	0	0
	4	0	0	0
Time slot Constraint Violations	1	0	0	0
	2	0.2	0	0
	3	0	0	0.4
	4	0	0	1.2
Average Car Occupancy	1	3.9	3.9	3.9
	2	3.8	3.8	3.8
	3	3.7	3.7	3.7
	4	3.6	3.5	3.3

Table 4. T-Test results comparing the fitness of the individuals that comprise the final populations produced with the deviation constraint set at 10% and 50% and then between 10% and 90%. The values returned suggest that the fitness produced with 1 and 4 work patterns are statistically significant.

Comparison (Deviation)	Site 3	Site 2	Site 1
10% - 50%	0.0001	0.0001	0.0001
10% - 90%	0.0131	0.0001	0.0001

Table 5. T-Test results comparing the fitness of the individuals that comprise the final populations produced with the work pattern variable set at 1 and 4. The values returned suggest that the fitness produced with 1 and 4 work patterns are statistically significant.

Comparison (Groups)	Site 3	Site 2	Site 1
1 - 4	0.0001	0.0001	0.0001

From an environmental perspective, it is interesting to note the average occupancy of the cars arriving at the workplace. An optimal solution would present an occupancy of 4, table 3 presents solutions close to this (an average of 3.9 and 3.8) when only 1 or two timeslot constraints exist.

Over the following series of the number of work patterns available was increased from 1 to 4 and results obtained may be seen in table 3. T-tests have been used in order to establish that varying the deviation constraint and the quantity of work patterns does result in statistically significant changes in results. Table 4 compares the fitness of the pupations produced with a deviation constraint of 10% with the results achieved with constraints of 50% and 90%. Table 5 makes a similar comparison between results obtained with only one work pattern and with 4 work patterns.

6 Conclusions and Future Work

From the results presented it may be seen that total commuter millage is reduced by 50%, and on average less than 30% of employees actually have to park at work. Given the constraint of limiting individuals to a maximum of 4 persons per vehicle, the algorithm manages to reduce the number of cars at the workplace to less than 1% more than the 25% minimum.

In every case individuals' desires for deviation distance were met and only in a few cases at the largest campus were some individuals not placed in groups compatible with their timeslot. This system produces a plan within approximately 10 minutes of CPU time, although this time will differ depending on hardware , software implementation and on the dataset being used. Future work, involves allowing a wider range of user variables to be taken into account, the constraining nature of such variables and their potentially random nature should create a search space that may be successfully explore using the evolutionary algorithm. The survey work undertaken in [2] suggests that there is potential for modelling of soft constraints. This may be achieved by allowing users to feedback into the system their satisfaction level with the arrangements proposed. Such feedback would allow individuals to be allocated reputational scores indicating their tolerance of sharing. It would be possible to build up a graph structure of employees with weighted arcs indicating that individuals have shared previously and the success of that share.

References

1. Buchholz, F.: The carpool problem. Technical report, Institute of Computer Science, University of Stuttgard (1997)
2. Holden, R., Urquhart, N., McEwan, T., Vogogias, T.: Co2y: the intelligent green solution: minimising carbon emissions by maximising shared travel opportunity. In: Scottish Transport Applications Research Conference, Glasgow, UK (2009)
3. Naor, M.: On fairness in the carpool problem. Journal of Algorithms 55(1), 93–98 (2005)
4. Ombuki, B.M., Ross, B., Hanshar, F.: Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl. Intell. 24(1), 17–30 (2006)
5. Roumboutsos, A., Kapros, S.: A game theory approach to urban public transport integration policy. Transport Policy 15(4), 209–215 (2008)
6. Runka, A., Ombuki-Berman, B.M., Ventresca, M.: A search space analysis for the waste collection vehicle routing problem with time windows. In: Rothlauf, F. (ed.) GECCO, pp. 1813–1814. ACM, New York (2009)
7. Toth, P., Vigo, D.: The Vehicle Routing Problem. Society for Industrial and Applied Mathematics (2002)
8. Urquhart, N.: Carbon-friendly travel plan construction using an evolutionary algorithm. In: Lipson, H. (ed.) GECCO, p. 2269. ACM, New York (2007)

A PSO-Based Memetic Algorithm for the Team Orienteering Problem

Duc-Cuong Dang[1], Rym Nesrine Guibadj[1,2], and Aziz Moukrim[1]

[1] Université de Technologie de Compiègne
Heudiasyc, CNRS UMR 6599, BP 20529, 60205 Compiègne, France
[2] VEOLIA Transport, MERCUR subsidiary
15, rue du Louvre, 75001 Paris, France
{duc-cuong.dang,rym-nesrine.guibadj,aziz.moukrim}@hds.utc.fr

Abstract. This paper proposes an effective Particle Swarm Optimization (PSO)-based Memetic Algorithm (PSOMA) for the Team Orienteering Problem (TOP). TOP is a particular vehicle routing problem whose aim is to maximize the profit gained from visiting clients while not exceeding a travel cost/time limit. Our PSOMA features optimal splitting techniques and genetic crossover operators. Furthermore, the memetic characteristic of our PSOMA is strengthened by an intensive use of local search techniques and also by a low value of 0.07 for inertia. In our experiments with the standard benchmark for TOP, PSOMA attained a gap of only 0.016%, as compared to 0.041%, the best known gap in the literature.

Keywords: Swarm Intelligence, Metaheuristics, Team Orienteering Problem, Optimal Split.

Introduction

The term Team Orienteering Problem (TOP), first introduced in [9], comes from an outdoor game played in mountainous or forested areas. In this game a team of several players try to collect as many reward points as possible within a given time limit. The Vehicle Routing Problem (VRP), analogous to the game that we denote simply by TOP, is the problem where a limited number of vehicles are available to visit customers from a potential set, the travel time of each vehicle being limited by a time quota, customers having different corresponding profits, and each customer being visited once at most once. The aim of TOP is to organize an itinerary of visits so as to maximize the total profit.

TOP is a variant of the Orienteering Problem (OP, also known as the Selective Traveling Salesman Problem) for multiple vehicles. As an extension of OP [10], TOP is clearly NP-Hard. OP and its variants have attracted a good deal of attention in recent years as a result of their practical applications and their hardness [5, 7, 9, 10, 14, 16, 20, 21, 22]. Readers are referred to [23] for a recent survey of these problems.

In this paper we are interested in TOP as the core variant of OP for multiple vehicles. This work was motivated by several lines of research first put forward

© Springer-Verlag Berlin Heidelberg 2011

by Veolia Environnement [4, 5]. Solving TOP to optimality has not received much attention. As far as we know, there are only two exact algorithms for TOP [6, 8]. Both are branch-and-price algorithms, but the second has the advantage of being easily adaptable to different variants of TOP. In contrast to exact solving approaches, a number of heuristics and metaheuristics have been developed for TOP [1, 4, 9, 12, 19, 20]. Three of these methods are considered to be state-of-the-art algorithms for TOP. The first is the slow version of Variable Neighborhood Search (SVNS) in [1]. The second is the Memetic Algorithm (MA) in [4], and the third is the slow version of Path Relinking approach (SPR) in [19].

The main contribution of this paper is a new memetic algorithm, called PSOMA, that can provide high quality solutions for TOP. The algorithm is relatively close to MA proposed in [4] and features the same basic components such as tour-splitting technique, population initializer and local search neighborhoods. However the global scheme has been changed to Particle Swarm Optimization (PSO): the recombination operator taking account of three sequences instead of two in MA and especially configurable with PSO parameters; a *swarm* of *particles*, i.e. couples of sequences, instead of a population of sequences in MA; a systematical application of recombination operator to every particle of the swarm in comparison to the stochastic selection of sequences for crossover operator in MA. With this memetic variant of PSO we were able to determine that good quality solutions require a very low value for *inertia*, and this can be attributed to the memetic characteristic of the algorithm. Experiments conducted on standard benchmark instances have shown clearly that with such an inertia, PSOMA was able to obtain better results than the state-of-the-art algorithms, including MA, with less computational effort.

The remainder of this paper is organized as follows. Section 1 provides a formal formulation of TOP. Our PSO-based Memetic Algorithm (PSOMA) is described in Section 2. Section 3 describes our empirical method for tuning PSOMA parameters and discusses computational results on benchmark instances. Finally, some conclusions and further developments are discussed in Section 4.

1 Formulation of the Problem

TOP is modeled with a graph $G = (V \cup \{d\} \cup \{a\}, E)$, where $V = \{1, 2, ..., n\}$ is the set of vertices representing customers, $E = \{(i, j) \mid i, j \in V\}$ is the edge set, and d and a are respectively departure and arrival vertices for vehicles. Each vertex i is associated with a profit P_i, and each edge $(i, j) \in E$ is associated with a travel cost $C_{i,j}$ which is assumed to be symmetric and satisfying the triangle inequality. A tour r is represented as an ordered list of $|r|$ customers from V, so $r = (i_1, \ldots, i_{|r|})$. Each *tour* begins at the departure vertex and ends at the arrival vertex. We denote the total profit collected from a tour r as $P(r) = \sum_{i \in r} P_i$, and the total travel cost/time as $C(r) = C_{d,i_1} + \sum_{x=1}^{x=|r|-1} C_{i_x,i_{x+1}} + C_{i_{|r|},a}$. A tour r is feasible if $C(r) \leq L$ with L being a predefined travel cost/time limit. The fleet is composed of m identical vehicles. A *solution* S is consequently a set of m (or fewer) feasible tours in which each customer is visited only once.

The goal is to find a solution S such that $\sum_{r \in S} P(r)$ is maximized. One simple way of reducing the size of the problem is to consider only *accessible* clients. A client is said to be accessible if a tour containing only this client has a travel cost/time less than or equal to L. For mixed integer linear programming formulations of TOP see [6, 8, 12, 23].

2 PSO-Based Memetic Algorithm

Particle Swarm Optimization (PSO) [13, 18] is one of swarm intelligence techniques with the basic idea of simulating the collective intelligence and social behavior of wild animals that can be observed, for example, in fish schooling and bird flocking. PSO was first used for optimization problem in continuous space as follows. A set known as a *swarm* of candidate solutions, referred to as *particles*, is composed of positions in the search space. The swarm explores the search space according to Equations 1 and 2. In these equations, x_i^t and v_i^t are respectively the position and the velocity of particle i at instant t. Three values w, c_1 and c_2, called respectively *inertia*, *cognitive* factor and *social* factor, are parameters of the algorithm. Two values r_1 and r_2 are random numbers generated in the interval $[0, 1]$. Each particle i memorizes its best known position up to instant t as x_i^{lbest}, and the best known position up to instant t for the swarm is denoted as x^{gbest}.

$$v_i^{t+1}[j] = w.v_i^t[j] + c_1.r_1.(x_i^{lbest}[j] - x_i^t[j]) + c_2.r_2.(x^{gbest}[j] - x_i^t[j]) \tag{1}$$
$$x_i^{t+1}[j] = x_i^t[j] + v_i^{t+1}[j] \tag{2}$$

With this design, PSO is highly successful at performing optimizations in continuous space [2, 11]. In contrast, when applied to problems of combinatorial optimization, PSO encounters difficulties in interpreting positions and velocities, as well in defining position update operators. As result, there are a variety of discrete PSO variants (DPSO) [3], and it is difficult to choose an appropriate variant for any given combinatorial optimization such as TOP.

Memetic algorithms (MA) [15] represent an optimization technique that attempts to simulate social evolution rather than genetic or biological evolution. Most MA designs incorporate various local search techniques into a global search scheme, e.g. a genetic algorithm. MA and PSO are both based on social evolution or behavior rather than biological ones, and there are benefits to be gained from combining techniques into a single form, that we call a *PSO-based MA*, or *PSOMA*, for solving combinatorial optimization problems. This section examines PSOMA in detail as a technique for solving TOP.

2.1 Position Representation and Improvement of Positions

A position in PSOMA is a permutation π of all accessible clients in a particular problem scenario. [4] gives a *splitting* algorithm that optimally transforms the permutation into a solution of TOP in $O(m.n^2)$. This algorithm guarantees that if the tours forming one of the optimal solutions of TOP are subsequences in a permutation π^*, the optimal solution will be returned in the output of the algorithm. This is made possible by considering only *saturated* tours respecting the permutation order, i.e subsequences of the permutation respecting the travel cost/time limit and containing the highest number of clients, then formulating the selection of the most profitable m saturated tours as a longest path problem under a k-cardinality constraint (kCC-LPP) on an auxiliary acylic graph. Here the value of k is $2.m+1$ and kCC-LPP can be solved efficiently through dynamic programming. In our approach, we use the PSO principle to find such a π^* permutation.

The authors of [4] also provided an efficient local search technique (LS) to improve a given permutation. Their intensive use of LS made the method a memetic method. Consequently, in our PSOMA, whenever a new position is found it has a pm probability of being improved using the same LS. It contains 3 neighborhoods:

– *shift operator*: evaluate all possibilities of moving a customer i from its original position to any other position in the permutation.
– *swap operator*: evaluate all possibilities of exchanging two customers i and j in the permutation.
– *destruction/repair operator*: evaluate the possibility of removing a random number (between 1 and $\frac{n}{m}$) of clients from an identified solution and then rebuilding the solution with a best insertion algorithm. Best insertion uses the well-known intuitive criterion for TOP that maximizes $\frac{\Delta P_i}{\Delta C_i}$ [4, 19].

The procedure is as follows. One neighborhood is randomly chosen to be applied to the particle position. As soon as an improvement is found, it is applied and the LS procedure is restarted from the new improved position. The LS is stopped when all neighborhoods are fully applied without there being any improvement.

2.2 Genetic Crossover Operator to Update Position

In combinatorial optimization, the particle position update of PSO can be interpreted as a recombination of three positions/solutions according to inertia, cognitive and social parameters. There are various ways of defining this kind of recombination operator [3]. In our approach, the recombination operator is similar to a genetic crossover whose core component is an extraction of l clients from a permutation π while avoiding clients from M set. This avoiding set allows extracted subsequences from successive calls of the core component to be non collapsed, so then they can be reassembled into a new valid sequence. The extracted subsequence is denoted π^l_M and the procedure is described as follows:

– Step 1: generate a random location r in π and initialize π^l_M to empty.
– Step 2: browse clients from $\pi[r]$ to $\pi[n]$ and add them to the end of π^l_M if they are not in M. The set of clients to be avoided M is updated during the process. If $|\pi^l_M|$ reaches l then terminate, otherwise go to Step 3.
– Step 3: browse clients from $\pi[r]$ down to $\pi[1]$ and add them to the beginning of π^l_M if they are not in M. The set of clients to be avoided M is updated during the process. If $|\pi^l_M|$ reaches l then terminate.

With the core component, the position update procedure of particle x from the swarm S with respect to the three PSO parameters w, c_1 and c_2 is described as follows. For convenience, the current, local best and global best positions of the particle are denoted respectively $S[x].pos$, $S[x].lbest$ and $S[best].lbest$:

- Phase 1: apply sequentially but in a random order the core component to extract subsequences from $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ with a common set of clients to be avoided M, initialized to empty. The desired numbers of clients to be extracted for $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ are respectively $w.n$, $(1-w).n.\frac{c_1.r_1}{(c_1.r_1+c_2.r_2)}$ and $(1-w).n.\frac{c_2.r_2}{(c_1.r_1+c_2.r_2)}$. Here r_1 and r_2 are real numbers whose values are randomly generated in the interval $[0,1]$ with a uniform distribution.
- Phase 2: link three extracted subsequences in a random order to update $S[x].pos$.

Our particle position update procedure therefore works with the standard PSO parameters w, c_1 and c_2, the only restriction being that w has to be in the interval $[0,1]$. Our PSOMA can be classified as PSO with position only, because no velocity vector is employed. It might be remarked that the core component was created to adapt to a linear permutation order, but it can easily be adapted to a circular order by changing Step 3.

2.3 Swarm Local Best Update

In some situations, PSO can be trapped in a local optimum, especially when all the local best positions of particles in the swarm are identical. To avoid this premature convergence, whenever a new position is found by a particle x in the swarm S, instead of updating $S[x].lbest$, the algorithm will search for an appropriate particle y in the swarm and update $S[y].lbest$. The update rule is similar to [17] but simplified:

1. The update procedure is applied if and only if the performance of new position $S[x].pos$ is better than the worst local best $S[worst].lbest$.
2. If there exists a particle y in the S such that $S[y].lbest$ is similar $S[x].pos$, then replace $S[y].lbest$ with $S[x].pos$.
3. If no such particle y according to Rule 2 exists, replace $S[worst].lbest$ with $S[x].pos$. Each successful application of this rule indicates that a new local best has been *discovered* by the swarm.

The similarity measure in Rule 2 is based on two criteria: the total collected profit and the travel cost/time of the identified solution. Two positions are said to be similar or identical if the evaluation procedure on these positions returns the same profit and a difference in travel cost/time that is less than δ. The implementation of the update rules was made efficient through the use of a binary search tree to sort particles by the performance of their local best positions using the two criteria.

2.4 Global PSO Algorithm

Particle positions in the swarm, including local best positions, are initialized to a random sequence. In order to accelerate the algorithm, a small portion of the swarm containing K particles will have local best positions generated with Iterative Destruction/Construction Heuristics (IDCH) described in [4]. PSOMA is stopped when the upper bound [6] is reached or after *itermax* consecutive iterations have failed to give rise to new local best. The global scheme is summarized in Algorithm 1.

```
Data: $S$ a swarm of $N$ particles;
Result: $S[best].lbest$ best position found;
begin
    initialize and evaluate each particle in $S$;
    $iter \leftarrow 1$;
    while $iter \leq itermax$ do
        foreach $x$ in $[1..N]$ do
            update $S[x].pos$ (see Section 2.2);
            if $rand(0,1) < pm$ then
                apply local search on $S[x].pos$ (see Section 2.1);
            end
            evaluate $S[x].pos$ (see Section 2.1);
            update $lbest$ of $S$ (see Section 2.3);
            if (a new local best is discovered) then
                $iter \leftarrow 1$;
            else
                $iter \leftarrow iter + 1$;
            end
        end
    end
end
```

Algorithm 1. Basic PSOMA scheme

3 Parameter Configuration and Numerical Results

Our algorithm is coded in C++ using the Standard Template Library (STL) for data structures. The program is compiled with GNU GCC in a Linux environment, and all experiments were conducted on an AMD Opteron 2.60 GHz. This section describes in detail our method for tuning PSOMA parameters, in particular the PSO inertia parameter. We also compare the performances of our approach with those of state-of-the-art algorithms in the literature, using 387 instances from the standard benchmark for TOP [9]. These instances comprise 7 sets. Inside each set the original number of customers and customer positions are constant, however the maximum tour duration L varies, then the number of accessible clients are different for each instance. The number of vehicles m also varies between 2 and 4.

3.1 Parameter Tuning

We decided to reuse most of the population management parameter values listed in [4]. To simplify the PSO parameter tuning, we decided to set the ratio between c_1 and c_2 to equality, meaning that a particle has the same probability of moving either to local or to global best. These parameters are summarized as follows.

– N, population size, is set to 40,
– K, the number of local best positions initialized with IDCH, is set to 5,
– $itermax$, the stopping condition, is set to $10 \cdot \frac{n}{m}$,
– pm, the local search rate, is set to $1 - \frac{iter}{itermax}$,
– δ, the similarity measurement of particles, is set to 0.01,
– c_1, the PSO cognitive factor, is set to 1.41,
– c_2, the PSO social factor, is set to 1.41.

The only remaining parameter is inertia w, with values from 0 to 1.0. According to [18] the choice of the inertia value is crucial to the performance of PSO. So two experiments were designed to determine the best value for w and also the best variant of the algorithm to use.

Our first experiment was designed to identify the prominent region for the value of w. The algorithms were tested with large steps between values of w, these values being 0, 0.1, 0.3, 0.5, 0.7 and 0.9. For each of these values the algorithms were executed 10 times for each instance of the benchmark. The sum of the highest profits for each instance could then be compared with the corresponding sum from the best known results in the literature. These best

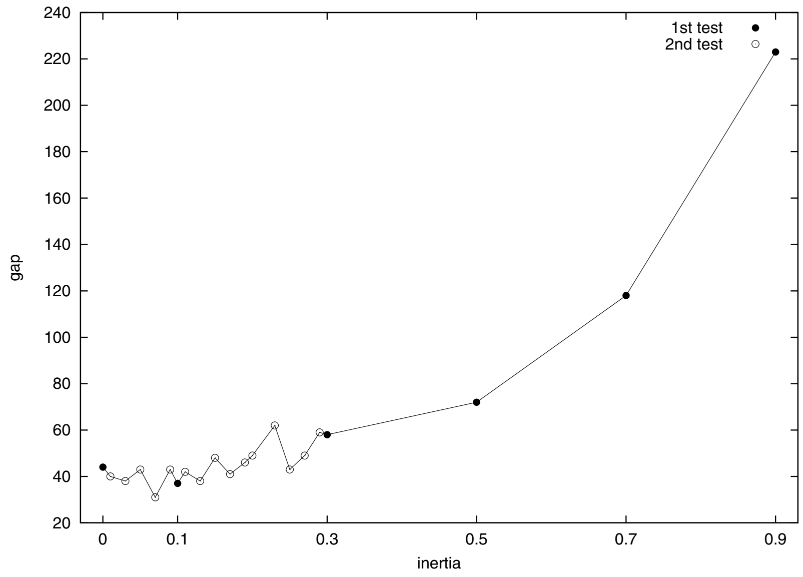


Fig. 1. Gap to the best known results for PSOMA with different values of w

Table 1. Comparison with state-of-the-art algorithms

Method	Parameter	Average CPU time on data sets							Gap	
		1	2	3	4	5	6	7	Profit	Percent
PSOMA	$w = 0.10$	0.14	0.01	0.51	82.94	12.81	3.87	55.98	37	0.019
	$w = 0.07$	0.15	0.01	0.51	78.45	12.87	3.99	54.54	31	0.016
SPR[a]	*slow*	–	–	–	367.40	119.90	89.60	272.80	117	0.060
MA[b]	$k = 5$	1.31	0.13	1.56	125.26	23.96	15.53	90.30	81	0.041
	$k = 10$	1.74	0.24	2.06	170.21	33.52	22.33	109.00	54	0.028
SVNS[c]	*slow*	7.78	0.03	10.19	457.89	158.93	147.88	309.87	85	0.043

[a] Computations were carried out on an Intel Xeon 2.50 GHz
[b] Computations were carried out on an Intel Core 2 Duo 2.67 GHz
[c] Computations were carried out on an Intel Pentium 4 2.80 GHz

results are collected from [1, 4, 12, 19, 20] but not from [9] because the authors used a different rounding precision and some of their results exceeded the upper bounds given in [6]. Here we are interested in the difference (or *gap*) between those two sums. On the basis of the results shown in Figure 1 for this experiment, we chose to perform the next test using different values of w in the prominent region between 0 and 0.3.

In the second test, PSOMA was executed, as a complement to the first test, with the following values for w: 0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.19, 0.20, 0.21, 0.23, 0.25, 0.27 and 0.29. Figure 1 gives a full graphical representation of the differences with the best known profits from the literature. The figure shows that the best value that we found for w is 0.07.

3.2 Numerical Comparisons with Existing Methods

Numerical results with comparison to the state-of-the-art algorithms, MA in [4], SVNS in [1] and SPR in [19], are given in Table 1. In this table, the relative gap in percent is the gap in profit over the sum of profits from the best known solutions in the literature. Results of SPR were not reported for all instances in [19], so CPU times of SPR for sets 1, 2 and 3 are not reported in the table. But according to the authors, the best known results in the literature can be used as score of SPR for unreported instances, hence its gap can be deduced.

After the first test we noticed that PSOMA with $w = 0.1$ already outperformed all the three state-of-the-art algorithms. MA, SVNS and SPR gave gaps of 81, 85 and 117 respectively in relation to the standard benchmark, while the gap given by PSOMA is 37. Even better results were obtained with $w = 0.07$, where the gap was reduced to 31. However, in order for the comparison with [4] to be fair, we obtained the source code from the authors and applied the same test protocol to MA with the stopping condition set to $10 \cdot \frac{n}{m}$, and with

10 executions per instance. These results are also shown in Table 1, the gap for MA obtained from this test was 54, rather than the original 81. Regarding computational time, PSOMA required less CPU time than MA.

4 Conclusion

This paper presented a novel memetic algorithm for the Team Orienteering Problem which incorporates the PSO principle into the main scheme. Numerical results on the standard benchmark for TOP demonstrate the competitiveness of this approach. The new PSOMA outperformed the prior GA/MA design in terms both of computation time and solution quality. Because the old GA/MA is one of the state-of-the-art algorithms, the new PSOMA has considerably improved the solving method for TOP, the newly attained gap being 0.016%. This success is due to the good design of the recombination operator for updating particle positions, as well as to an appropriate choice of parameters. Furthermore, a small inertia value, corresponding to the best configuration of the algorithm, strengthens the memetic characteristic of the approach. In summary, the results presented in this paper are encouraging for the application of Particle Swarm Optimization for solving combinatorial problems, as has already been indicated in [3]. As part of our future development of this method we intend to investigate the performance of the approach for certain variants of TOP, such as variants with time windows or with capacity constraints.

References

[1] Archetti, C., Hertz, A., Speranza, M.: Metaheuristics for the team orienteering problem. Journal of Heuristics 13(1) (February 2006)
[2] Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. part I: background and development. Natural Computing 6(4), 467–484 (2007)
[3] Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing 7(1), 109–124 (2008)
[4] Bouly, H., Dang, D.C., Moukrim, A.: A memetic algorithm for the team orienteering problem. 4OR 8(1), 49–70 (2010)
[5] Bouly, H., Moukrim, A., Chanteur, D., Simon, L.: Un algorithme de destruction/construction itératif pour la résolution d'un problème de tournées de véhicules spécifique. In: MOSIM 2008 (2008)
[6] Boussier, S., Feillet, D., Gendreau, M.: An exact algorithm for team orienteering problems. 4OR 5(3), 211–230 (2007)
[7] Butt, S., Cavalier, T.: A heuristic for the multiple tour maximum collection problem. Computers and Operations Research 21, 101–111 (1994)
[8] Butt, S.E., Ryan, D.M.: An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers and Operations Research 26, 427–441 (1999)
[9] Chao, I.M., Golden, B., Wasil, E.: The team orienteering problem. European Journal of Operational Research 88 (1996)

[10] Golden, B., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics 34, 307–318 (1987)
[11] Kameyama, K.: Particle swarm optimization - a survey. IEICE Transactions 92-D(7), 1354–1361 (2009)
[12] Ke, L., Archetti, C., Feng, Z.: Ants can solve the team orienteering problem. Computers and Industrial Engineering 54(3), 648–665 (2008)
[13] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
[14] Montemanni, R., Gambardella, L.: Ant colony system for team orienteering problems with time windows. Foundations of Computing and Decision Sciences 34, 287–306 (2009)
[15] Moscato, P.: Memetic Algorithms: a short introduction. In: New Ideas in Optimization, pp. 219–234. McGraw-Hill, New York (1999)
[16] Schilde, M., Doerner, K.F., Hartl, R.F., Kiechle, G.: Metaheuristics for the bi-objective orienteering problem. Swarm Intelligence 3(3), 179–201 (2009)
[17] Sha, D.Y., Hsu, C.Y.: A hybird particle swarm optimization for job shop scheduling problem. Computers and Industrial Engineering 51(4), 791–808 (2006)
[18] Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Evolutionary Programming, pp. 591–600 (1998)
[19] Souffriau, W., Vansteenwegen, P., Van den Berghe, G., Van Oudheusden, D.: A path relinking approach for the team orienteering problem. Computers & Operations Research 37(11), 1853–1859 (2010)
[20] Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computer & Operations Research 32, 1379–1407 (2005)
[21] Tricoire, F., Romauch, M., Doerner, K.F., Hartl, R.F.: Heuristics for the multi-period orienteering problem with multiple time windows. Computers & Operations Research 37, 351–367 (2010)
[22] Vansteenwegen, P., Souffriau, W., Van den Berghe, G., Van Oudheusden, D.: Metaheuristics for tourist trip planning. In: Metaheuristics in the Service Industry. LNEMS, vol. 624, pp. 15–31. Springer, Heidelberg (2009)
[23] Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: A survey. European Journal of Operational Research 209(1), 1–10 (2011)

Heuristics for a Real-World Mail Delivery Problem

Elisabeth Gussmagg-Pfliegl[1], Fabien Tricoire[1], Karl F. Doerner[1,2], Richard F. Hartl[1], and Stefan Irnich[3]

[1] University of Vienna
[2] Johannes Kepler University Linz
[3] Johannes Gutenberg University Mainz

Abstract. We are solving a mail delivery problem by combining exact and heuristic methods. The problem is a tactical routing problem as routes for all postpersons have to be planned in advance for a period of several months. As for many other routing problems, the task is to construct a set of feasible routes serving each customer exactly once at minimum cost. Four different modes (car, moped, bicycle, and walking) are available, but not all customers are accessible by all modes. Thus, the problem is characterized by three interdependent decisions: the clustering of customers into districts, the choice of a mode for each district, and the routing of the postperson through its district. We present a two-phase solution approach that we have implemented and tested on real world instances. Results show that the approach can compete with solutions currently employed and is able to improve them by up to 9.5%.

Keywords: mail delivery, hybrid solution approach, set covering.

1 Motivation and Introduction

Every day, millions of private households are waiting for their mail in European countries. Currently, in many countries only public companies are responsible for delivering the mail. After deregulation of the postal services by the European Union, also private companies are allowed to deliver letters to households. The current public actors, as well as potential private actors, are expecting high competition for this market and, therefore, seek to improve their competitiveness.

Regardless of the changing market environment, postmen routes need to be adapted on a regular basis because customer demand is changing over time (e.g. new houses are being built in the area, people are moving to another area, etc.). These adaptations can be minor or major adjustments (Bodin and Levy 2000). On the one hand, making only minor changes to existing routes means ignoring potential savings, but can be implemented very quickly. On the other hand, the planning and implementation of completely new routes is quite costly and is therefore omitted sometimes.

Orloff (1974) recommends an arc routing formulation for mail delivery problems. The Capacitated Arc Routing Problem (CARP) consists of traversing arcs

© Springer-Verlag Berlin Heidelberg 2011

with positive demand with one of m identical vehicles so that cost is minimized. Arcs without demand can be traversed to reach compulsory ones. An introduction to the CARP can be found in the PhD thesis of Wøhlk (2006), where also some practical applications are described. She proposes a combination of dynamic programming and simulated annealing to solve the problem. On the other hand, Oppen and Løkketangen (2006) observe that, for a particular type of problem with a lot of customers located along streets, it is not obvious if a node routing or an arc routing formulation is more appropriate. This is also true in our case. Despite the fact that customers can be aggregated (as described in Sect. 2), a node routing environment is chosen, because we believe that, this way, we can solve the large problem instances we are facing more efficiently. The Vehicle Routing Problem (VRP) is similar to the CARP, but here demands are associated to nodes instead of arcs. The VRP is one of the most studied problems in the literature. An introduction to the VRP can be found in Toth and Vigo (2002) and in Golden et al. (2008). Recently, some rich VRPs have been introduced where real world constraints extend the basic VRP in several ways. As described by Doerner and Schmid (2010), matheuristics are a promising way to tackle rich VRPs. Here, exact and heuristic methods are combined to overcome the weaknesses of both methods. We will follow this observation and combine heuristics for generating a large pool of routes, and then solve a set covering problem to optimality to determine the most promising routes for the final solution. One of the first approaches using a similar method was published by Renaud et al. (1996). The authors generate a pool of routes using an improved petal heuristic, and then solve a set partitioning problem to determine an optimal combination of routes. Recently, Muter et al. (2010) proposed the integration of an exact algorithm into a tabu search framework. The exact method solves a set covering problem and guides the metaheuristic in the solution space.

The remainder of the paper is organized as follows. A detailed problem description is given in Sect. 2. The solution approach is described in Sect. 3, and a case study is presented in Sect. 4. A summary concludes the paper in Sect. 5.

2 Problem Description

We are planning mail delivery tours where each tour starts and ends at a given depot. A tour is performed by a postperson, and each person may either use a vehicle of the depot fleet to perform the tour, or walk. A set of customers is provided, and each customer has to be served exactly once. Since it is reasonable to visit groups of customers consecutively (i.e., all customers living in a certain part of a street), we aggregate customers to street segments that are then treated as a single task. This helps to reduce the problem size because only entry and exit points of the street segments, called connecting points, are considered. How the customers are aggregated depends on the characteristics of the street they live in. While in small streets customers can be served in a zigzag pattern, visiting both street sides at the same time, in larger streets each side has to be served separately, to reduce the risk of road accidents for postpersons. Figure 1 shows

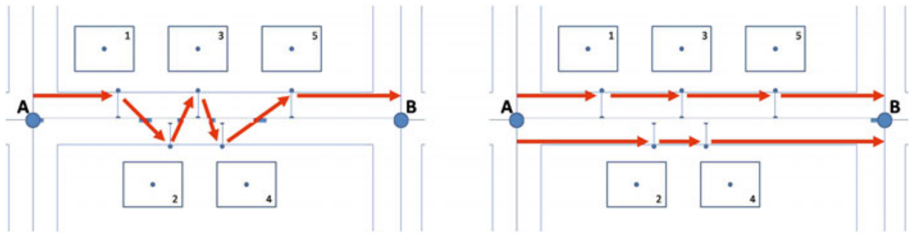


Fig. 1. If a street can be served in a zigzag pattern (shown on the left), then a single street segment represents the customers in this street. If the street has to be served side by side, then two independent street segments are created, as shown on the right.

examples of possible walking patterns. Whether a street can be served using a zigzag pattern or not is decided beforehand by the mail delivery company and is not a decision we have to make in the solution process. For some streets (i.e. one way streets), service is only possible in the direction of traffic, but many streets can be served in both directions. The proportion of one way streets varies from instance to instance, since urban as well as rural areas are considered. In urban areas, many one way streets can be found, whereas in rural areas most streets are bi-directional. Using walking mode, all streets can be served in both directions. Each street segment may be traversed an arbitrary number of times without servicing it to reach other segments, which is often called deadheading.

Fleet. The depot fleet consists of bikes, mopeds, and cars. In addition, a postperson might walk and use a hand cart to carry the mail. For each type of vehicle (or mode, as we refer to it here), the amount of vehicles available is specified. For each type, vehicles are assumed to be homogeneous (i.e. all vehicles of a type have the same capacity and the same speed), and each postperson is able to use any of the vehicles available. Each vehicle used causes a fixed amount of maintenance cost.

Postpersons. The different contracts of postal employees determine the person's daily working time and the amount of extra working hours (overtime) that might be added if necessary. For each contract type, the number of available persons is known and must be respected in the solution. In addition, the mail has to be prepared before starting the tours.This is called table work and consists of sorting the letters in the order of delivery in the tour. This work can either be done by the postperson performing the tour or by another employee. In some cases it is necessary to separate the table work from the delivery task. Whether a postperson performs the table work himself or not is decided beforehand by the company. The amount of time needed to sort the mail depends on the street segments visited in the tour. In case the performing postperson sorts the mail, the table work is part of the total working time and therefore reduces the tourlength (the sum of both must not exceed the person's maximum working time). Before a postperson starts the tour, he/she fills the vehicle (or the hand cart if it is a walking tour) with the sorted mail. If the available capacity is too small, the remaining mail is taken to relay boxes along the route by another worker. If necessary, a postperson's working time can be extended. The maximum amount of overtime depends on the personnel type.

Network. We use a directed street network to reflect the real street network and to take one way streets and turning restrictions into account. For street segments that have to be served, the connecting points are known, as well as cost and time needed to serve the segment. Note that time and cost depend on the vehicle (mode) used to traverse an arc, which means that in total up to eight different costs can be assigned to a single arc. (Two values for each mode: with and without service. The service cost of a street segment depends on the vehicle used. Therefore, the decision of which mode should be used to service a segment has strong impact on the objective value. In a similar fashion, service times also depend on the vehicles used. The vehicle selection influences the route duration and, therefore, the route feasibility.)

ZIP-codes are an important issue in the planning of postmen tours, because machines are used to pre-sort the mail. The more ZIP-code borders a route crosses, the more difficult it becomes to automatically sort the letters. Therefore, ZIP-code boundaries are also included in the network by adding a penalty to the costs of arcs that are crossing a border.

Objective. The objective is to minimize total cost, which consists of the service cost for each street segment, the routing cost in between segments, and the cost for preparing the mail for the covered segments in the morning. For each vehicle used in the solution, a small amount of maintenance cost (depending on the type of vehicle) is also added. Similar to Bodin et al. (2000), routes should reach as close as possible the maximum working time of the performing postperson because of practical considerations. Since the company could not specify this explicitly enough to formulate it in a mathematical way, we do not consider this aspect in the objective function, but pay attention to it in the final steps of the solution procedure.

3 Solution Approach

The solution approach consists of two steps (see Fig. 2). First, a clustering and routing process produces to a pool of routes. Then, in the second step, a set covering problem is solved where routes are chosen from the pool to form the final solution. Both steps are explained in detail in the following.

3.1 Step One: Clustering and Routing

In step one, routes are generated that serve as input for the set covering problem solved in step two. To form homogeneous routes, we follow the cluster-first route-second paradigm proposed by Fisher and Jaikumar (1981).

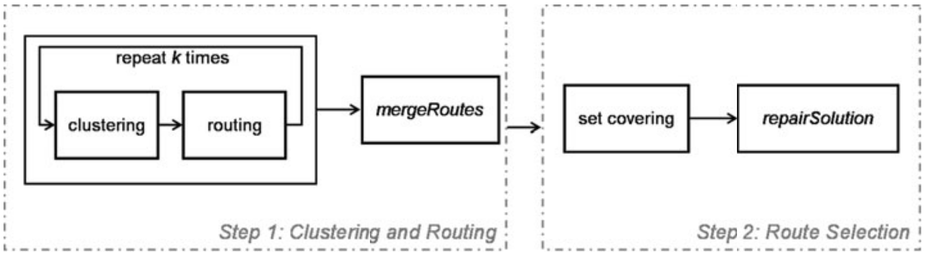


Fig. 2. Steps in the solution approach

Clustering Algorithm. Clusters are formed by placing seeds in a heuristic way. The first seed is chosen randomly, then the node farthest away from the seed is chosen, then the node farthest away from all seeds chosen so far, and so on. The generation of seeds stops when the number of seeds equals the number of postpersons available or when no more nodes are available. This is the case if all points can be reached from a seed within distance $Dmin$ (see Algorithm 1, $Dmin$ is a parameter).

Once all seeds are determined, each of the remaining nodes is assigned to its nearest seed. One cluster is formed for each seed and its assigned nodes. Then, for each cluster, routes are generated.

Algorithm 1. Generation of Seeds for Clusters

```
$Seeds \leftarrow \{randomElement(\mathcal{N})\}$
$Candidates \leftarrow \mathcal{N} \setminus Seeds$
**repeat**
    $s \leftarrow$ farthest element from all $Seeds$ in $Candidates$
    $Seeds \leftarrow Seeds \cup \{s\}$
    $Candidates \leftarrow Candidates \setminus \{i | d_{is} < D_{min}\}$
**until** $|Seeds| = nClustersMax \vee Candidates = \emptyset$
return $Seeds$
```

Routing Algorithm. We repeat the clustering for each personnel type and each mode available. Therefore each cluster has a personnel type associated to it. This is important since the postperson's maximum working time per day limits the route length, and the working times of postpersons vary because of different working contracts. Routes are generated using the well known savings algorithm by Clarke and Wright (1964). Starting from the depot, for each street segment, the cheapest route to serve just this single segment is determined. The route through each segment is fixed so that savings can be calculated. We use the parallel version of the algorithm and merge routes in the order of decreasing savings if it is feasible w.r.t route duration, until no more routes can be merged. Then the generated routes are added to the pool and the clustering starts again for the next postperson.

mergeRoutes. The Savings Algorithm tends to produce routes that are much shorter than the maximum duration. Therefore, after the route generation is finished for all postpersons and modes, we try to merge routes again. This time, we consider all routes in the pool. Two routes r and l in the pool are merged if both routes are using the same mode, and the route duration of the combined routes does not exceed the maximum route duration of the postperson performing route r. If both conditions are satisfied, customers in route l are appended to route r and route l is removed from the pool of routes. Since the pool is very large, we do not consider all possibilities but apply the first profitable merge for each route.

3.2 Step Two: Route Selection

In the previous step, a number of routes were generated for each postperson using different modes. Now we need to select a set of routes that covers each street segment exactly once. Therefore a set covering problem is solved. To make sure that neither the number of persons available nor the number of vehicles available is exceeded, we introduce the concept of *resource.* A resource can be a person performing the route, or a vehicle. Note that a tour consumes several resources as each tour requires a person and a vehicle. The set covering model can be formulated as follows:

Ω ... set of all routes $j \in \Omega$
c_j ... cost of route j
u_j^r... amount of resource r consumed by route j
Q^r... available amount of resource r

$$a_j^i = \begin{cases} 1, & \text{if street segment } i \text{ is covered by route } j \\ 0, & \text{otherwise} \end{cases}$$

$$x_j = \begin{cases} 1, & \text{if route } j \text{ is used in the solution} \\ 0, & \text{otherwise} \end{cases}$$

$$min \quad \sum_{j \in \Omega} c_j x_j \tag{1}$$

$$\sum_{j \in \Omega} a_j^i x_j \geq 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{j \in \Omega} u_j^r x_j \leq Q^r \qquad \forall r \in R \tag{3}$$

The objective is to minimize total cost, which is the sum of costs of all selected subsets. Constraints (2) make sure that each street segment is covered at least once, while constraints (3) ensure that no more resources are used than available. Due to the set covering formulation, some street segments may be covered more than once. Since each segment has to be served exactly once, multiple visits are removed using the *repairSolution*-algorithm described in the following.

Local Search - *repairSolution* and *fillRoutes*. For each segment served more than once, the gain for removing this segment from the route is calculated for all routes serving this segment, and all visits are removed except the one with lowest gain. Afterwards, all routes are improved using a slightly modified version of the 3-opt* method proposed by Baker and Schaffer (1986).

For postal companies it is important that the route duration meets the maximum working time as close as possible. Therefore we use a local search called *fillRoutes* that tries to fill up the routes, and at the same time reduces the total number of routes. Even though the number of routes is not an objective in the first place, a solution with fewer routes decreases the fixed cost of the company and is therefore preferable. The algorithm works the following way: for each route, the difference between current and maximum route duration is calculated, and the route with the largest difference is chosen as candidate to be removed. The street segments of the candidate route are inserted into the remaining routes using a cheapest insertion criterion. If this leads to a feasible solution, we keep the new solution, remove the candidate and repeat the procedure again. Otherwise we keep the old solution and try to remove another route. The *fillRoutes* algorithm is terminated after ti iterations or if we fail k times in a row to insert all street segments of the candidate route into another one (ti and k are parameters). Then the last feasible solution is kept as final solution.

4 Case Study

Seven real world instances were used to test the algorithm. The instances are generated from rural, urban and mixed postal areas with different sizes. The number of street segments ranges between 895 in the smallest and 5,740 in the largest instance. For all instances, the average of three test runs is indicated in Table 1. Column *Segments* specifies the total number of street segments, column *Cost* provides the cost of the final solution. Column *Free Time* specifies the average difference between actual and maximum route duration for all routes in the solution. Column *Usage* indicates how much of the maximum route duration was used on average. Finally, the number of routes and the number of vehicles used in the solution are specified. (The number of vehicles is specified separately for each mode: walking/bike/moped/car).

Results were calculated on an Intel Xeon CPU with 2.67 GHz and 24 GB of RAM using the Linux operating system. The set covering model was solved using CPLEX 12.1 and was restricted to at most ten hours of computing time. This time limit was reached by only one instance (instance 6), which can be explained by the size of the set covering instance (4,228 rows and 23,000 columns, whereas for instance 7 only 18,600 columns were generated.) Instance 6 also had the longest overall computing time of approximately 59 hours. The smallest instance (instance 1) took less than 3 seconds to be solved, whereas the largest one (instance 7) was solved in 16.5 hours.

In the solutions in Table 1, the number of mopeds used as route mode often exceeds the number of the other vehicles. Even though this might be cheap, postal companies often avoid to use many mopeds because it is inconvenient and sometimes dangerous for employees during the year (rain in spring and fall season and snow in winter make). Therefore, we run a test where mopeds are forbidden. The results are indicated in Table 2. As expected, costs are slightly higher than before, and the total number of routes increases for all instances. Not all mopeds can be replaced by cars, therefore the number of bikes increases significantly. Bikes are considered less dangerous than mopeds because they drive with lower speed, but the company could also limit the number of bikes if necessary.

A third test was performed where the table work is not done by the postpersons themselves (see Table 3). As expected, fewer routes are needed to cover all street segments. The cost also reduces for all instances except instance 3, where it is slightly higher. This can probably be explained by the heuristic nature of the route generation procedure.

Table 1. Test instances and results

Instance	Segments	Cost	Free Time (min.)	Usage (%)	Number of Routes	Vehicles w/b/m/c
1	895	1,833.73	27.34	94.31	13.67	2.33/0/11.63/0
2	1110	3,640.30	40.54	91.56	23.33	14/0/7.67/1.67
3	1240	2,740.00	33.87	92.94	18.67	6/0/12.67/0
4	1676	2,287.13	53.05	88.95	15.67	3/0/10/2.67
5	1713	6,168.32	41.66	91.32	42	24/0.33/11/6.67
6	4228	7,933.97	15.66	96.74	55	14.67/1.33/31/7.67
7	5740	8,066.42	39.28	91.81	48	11.33/0/27.67/9

Table 2. Test run without mopeds

Instance	Segments	Cost	Free Time (min.)	Usage (%)	Number of Routes	Vehicles w/b/m/c
1	895	2,013.89	33.62	92.99	15.33	2/8.33/0/5
2	1110	4,136.84	82.87	82.74	29.67	16/8/0/6.67
3	1240	3,811.09	12.99	97.29	25	8.67/16.33/0/0
4	1676	2,434.69	20.78	95.67	16	2.67/6.33/0/7
5	1713	6,786.38	40.70	91.52	46.67	25.33/11.67/0/9.67
6	4228	8,332.53	10.15	97.88	57.67	16.33/15.67/0/25.67
7	5740	8,677.25	58.24	87.87	56	12/15/0/29

Table 3. Test run without table work

Instance	Segments	Cost	Free Time (min.)	Usage (%)	Number of Routes	Vehicles w/b/m/c
1	895	1,829.44	29.70	93.81	10.33	3/0/7.33/0
2	1110	3,578.50	77.62	83.83	22.33	13.67/0/6.67/2
3	1240	2,756.45	33.37	93.05	15.33	6/0.33/9/0
4	1676	2,120.25	61.11	87.27	12	2/0/7.33/2.67
5	1713	5,785.61	44.19	90.79	33.33	20/0/11/2.33
6	4228	7,309.68	24.46	94.90	39.67	6.33/0.67/27/2.33
7	5740	7,768.97	33.69	92.98	41	10.33/0/24.33/6.67

Finally, we compare our results to currently applied solutions. The comparison shows that an improvement of up to 9.5% can be reached, whereas in one case our solution is 5.9% more expensive. On average, a cost reduction of 3% can be obtained. To postal companies often not only the cost, but also the 'compactness' of routes is important for organizational reasons but very difficult to obtain using conventional routing algorithms. Because the company was not able to formulate this 'compactness' in a mathematical, company planners were asked to assess the obtained solutions and were satisfied with the presented results.

5 Summary and Conclusion

We presented a new approach to solve a tactical mail delivery problem. The complexity of the planning tasks results from large-scale instances with sometimes more than 5,000 customers and several non-standard options and constraints. We propose a cluster-first route-second approach where first a pool of promising routes is generated heuristically, and then a set covering problem is solved to optimality. Results on real world instances show that solutions can be implemented in practice and outperform existing solutions. However, we still see room for improvements, e.g., by applying more sophisticated routing algorithms in the first phase or by embedding the entire two-phase approach into an iterative framework that uses information from the set covering phase to construct improved clusters and routes.

Acknowledgements

Financial support from the Austrian Research Promotion Agency (FFG) under grant 821971 is gratefully acknowledged.

We would like to thank Patrick Blaim (myVision) and Nik Widmann (Prisma Solutions) for providing problem data from their industrial partner.

References

Baker, E.K., Schaffer, J.R.: Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. Amer. J. Math. Management Sci. 6, 261–300 (1986)

Bodin, L., Levy, L.: Scheduling of Local Delivery Carrier Routes for the United States Postal Service. In: Dror, M. (ed.) Arc Routing: Theory, Solutions, and Applications, pp. 419–442. Kluwer, Boston (2000)

Clarke, G., Wright, J.W.: Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research 12, 568–581 (1964)

Doerner, K.F., Schmid, V.: Survey: Matheuristics for Rich Vehicle Routing Problems. In: Blesa, M.J., Blum, C., Raidl, G., Roli, A., Sampels, M. (eds.) HM 2010. LNCS, vol. 6373, pp. 206–221. Springer, Heidelberg (2010)

Fisher, M.L., Jaikumar, R.: A generalized assignment heuristic for vehicle routing. Networks 11, 109–124 (1981)

Golden, B., Raghavan, S., Wasil, E. (eds.): The Vehicle Routing Problem. Latest Advances and New Challenges. Springer Science+Business Media, LLC (2008)

Muter, I., Birbil, S.I., Sahin, G.: Combination of Metaheuristic and Exact Algorithms for Solving Set Covering-Type Optimization Problems. Informs Journal on Computing 22(4), 603–619 (2010)

Oppen, J., Løkketangen, A.: Arc routing in a node routing environment. Computers & Operations Research 33, 1033–1055 (2006)

Orloff, C.S.: A Fundamental Problem in Vehicle Routing. Networks 4, 35–64 (1974)

Renaud, J., Boctor, F.F., Laporte, G.: An Improved Petal Heuristic for the Vehicle Routeing Problem. Journal of the Operational Research Society 47, 329–336 (1996)

Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (2002)

Wøhlk, S.: Contributions to Arc Routing. PhD thesis, University of Southern Denmark (2006)

Integrated Generation of Working Time Models and Staff Schedules in Workforce Management

Volker Nissen[1], Maik Günther[1], and René Schumann[2]

[1] TU Ilmenau, Information Systems in Services, D-98684 Ilmenau, Germany
volker.nissen@tu-ilmenau.de, maik.guenther@gmx.de
[2] University Frankfurt/M., Information Systems and Simulation, D-60325 Frankfurt, Germany
reschu@informatik.uni-frankfurt.de

Abstract. Our project addresses the question how to automatically and simultaneously assign staff to workstations and generate optimised working time models on the basis of fluctuating personnel demand while taking into account practical constraints. Two fundamentally different solution approaches, a specialized constructive heuristic (commercial) and a hybrid metaheuristic (the evolution strategy) that integrates a repair heuristic to remove contraint violations are compared on a complex real-world problem from a retailer. The hybrid approach clearly outperforms the tailored constructive method. Taken together with our similar findings on a related staff scheduling problem from logistics this result suggests that the evolution strategy, despite its original focus on continuous parameter optimisation, is a powerful tool in combinatorial optimisation and deserves more attention. Moreover, hybridising a metaheuristic with a problem-specific repair heuristic seems a useful approach of resolving the conflict between domain-specific characteristics of a real-world problem and the desire to employ generic optimisation techniques, at least in the domain of workforce management.

Keywords: integrated planning, metaheuristic, constructive heuristic.

1 Introduction

The ability to adapt the assignment of personnel to changing requirements is critical in workforce management (WFM). For the retail industry often inflexible shift models are used, thus leaving out an important option to increase WFM agility. In this traditional planning first a shift plan is created. Ernst et al. [5] describe this as the *line of work construction*. Staff scheduling as the next planning step involves the assignment of an appropriate employee to the appropriate workstation at the appropriate time while considering various constraints. This step is also referred to as *staff assignment* [5]. This traditional multi-level approach to workforce management in separate steps can be very inefficient.

An integrated design of working time models and staff schedules allows for better solutions. During this process, working time models are automatically

© Springer-Verlag Berlin Heidelberg 2011

generated based on demand while respecting certain constraints. Thereby, over- and understaffing are reduced while maintaining the same number of employees. Efficient usage of the workforce leads to a reduction of overtime and idle time, a rise in employee motivation and, thus, an increase of turnover through a higher level of service.

In this paper, we compare a constructive approach, specifically designed for the requirements of retail, and an adapted version of the evolution strategy on this task. While constructive approaches are generally popular in personnel scheduling and rostering [4], the evolution strategy performed well in a related scheduling problem from logistics [9].

The research goals discussed in this paper are twofold. First, we investigate means for finding good solutions to a complex practical application that is of relevance in such diverse industries as logistics, retail and call centres. Second, we want to contribute to the evaluation and comparison of the evolution strategy on combinatorial problems of realistic size and complexity, since this metaheuristic has received relatively little attention in combinatorial optimisation.

The rest of this article is structured as follows. In Section 2 the retail case is highlighted. Then we discuss related work. In Section 4 the constructive heuristic is presented. The subsequent section introduces our adaption of the evolution strategy. In Section 6 both heuristics are applied and the results are discussed. We conclude with a summary and indications for future work.

2 Description of the Real-World Case Study from a Retailer

This practical case concerns personnel planning in the department for ladies' wear at a department store. For current benchmarks and real data reference is made to [11]. In this problem we assume a set of nine employees $\mathcal{E} = \{1, \ldots, 9\}$, a single workstation $\mathcal{W}$ and a discrete timeframe $\mathcal{T}$ with index $t = 0, \ldots, T - 1$. Each period t of the interval has a length l_t greater than zero.

$$l_t > 0 \quad \forall t \in \mathcal{T} \tag{1}$$

The description of the workstation is kept rather open so that secondary tasks such as refilling can be integrated into the personnel demand. The assignment of an employee to the workstation is controlled using the binary variable x_{et}. A dummy workstation is used to mark times when an employee is generally available but is not dispatched in staffing. This approach is necessary due to the automatic generation of working time models within the scheduling framework.

$$x_{et} = \begin{cases} 1 & \text{if employee } e \text{ is assigned to the workstation at period } t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The store is open from Monday to Saturday from 10:00 to 20:00. Moreover, it is known in advance, which employees are on holiday or detained due to training.

The availability of the employees is determined using the binary variable a_{et}.

$$a_{et} = \begin{cases} 1 & \text{if employee } e \text{ is available at period } t \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The personnel demand d_t is given in one-hour intervals and is centrally determined based on historic data. Also considered are holidays as well as sales campaigns. A minimal and maximal number of employees per period is set.

In order to find a solution to the problem described, automatic working time model generation will be integrated with personnel scheduling. As a hard constraint, working time models must begin and end on the hour. Moreover, an employee e can only be associated with the workstation in the period t if he is actually present.

$$x_{et} \leq a_{et} \qquad \forall e \in \mathcal{E} \text{ and } \forall t \in \mathcal{T} \tag{4}$$

Additionally a number of soft constraints are introduced to model further requirements of the respective company. Their violation is penalised with error points that reflect their relative importance as inquired through interviews with the management. As maintaining a good service level is key for revenue in trade the violation of staffing targets has to be avoided. If a discrepancy arises between the staffing demand d_t and the assigned workforce d'_t, error points P_d are generated for the duration and size of the erroneous assignment. The assigned workforce d'_t is defined as follows.

$$d'_t = \sum_{e \in \mathcal{E}} x_{e,t} \tag{5}$$

Different types of errors can be distinguished: c_{do} represents overstaffing when the demand $d_t > 0$, c_{dn} signals overstaffing when the demand $d_t = 0$, c_{du} signals cases of understaffing.

$$P_d \;=\; \sum_{t=0}^{T-1} (c_{dn} + c_{do} + c_{du}) l_t \left| \left(\sum_{e=1}^{E} x_{et} \right) - d_t \right|,$$

with: (6)

$$c_{do} > 0 \text{ if the workstation is overstaffed at } t \text{ and } d_t > 0, \text{ else } c_{do} = 0,$$
$$c_{dn} > 0 \text{ if the workstation is overstaffed at } t \text{ and } d_t = 0, \text{ else } c_{dn} = 0,$$
$$c_{du} > 0 \text{ if the workstation is understaffed at } t \text{ and } d_t > 0, \text{ else } c_{du} = 0.$$

By assigning more error points in cases of overstaffing without demand c_{dn} the chance for mutual support of employees in their work is strengthened.

Four types of employment contracts exist. They differ in the amount of weekly working time which is between 25 and 40 hours. During weeks with bank holidays the planned working time s_e is reduced by a proportional factor h. The effective weekly working time i_e for an employee should not exceed the contractually agreed number of hours. Each minute in excess is punished with error points c_w.

$$P_w = \sum_{week=1}^{52} \sum_{e=1}^{E} c_w (i_e - s_e * h), \tag{7}$$

with: $c_w = 0$ if $s_e * h - i_e \geq 0$, $c_w = 1$ else.

A particular focus is on the automatically generated working time models. Working time models should not be shorter than 3 hours or longer than 9 hours. Any violation leads to error points c_t per employee and day. The sum of these error points for the planning horizon is P_t. Working time models must not be split up during a working day, with violations leading to error points c_c per employee and day. The sum of these error points for the planning horizon is P_c.

Therefore, the objective function to be minimised becomes:

$$minP = P_d + P_w + P_t + P_c. \tag{8}$$

Historical data is available for a complete calendar year, so that an entire year can be planned ahead. With 9 employees and 8.760 one-hour time slots to be planned the total number of decision variables for this problem is 78.840. In practice, also shorter planning horizons are employed. However, the full year plan helps the company to better understand on a more strategic level how well it can cope with demand using the current staff.

3 Related Work

Staff scheduling is a hard optimisation problem. Garey and Johnson [7] demonstrate that even simple versions of staff scheduling are NP-complete. Moreover, Tien and Kamiyama [23] prove that practical personnel scheduling problems are generally more complex than the TSP which is itself NP-hard.

The general workforce scheduling problem has attracted researchers for quite a number of years. Ernst et al.[4] provide a comprehensive overview of problems and solution methods for personnel scheduling and rostering from more than 700 analysed sources. Workforce scheduling problems can range from simple models like days-off scheduling to quite complex problems like scheduling in call centres [18] or multi objective optimisation problems [2]. Typical application areas are airline crew scheduling (see [5] for a survey), nurse scheduling in hospitals [3], and field workers scheduling (like repair service [13]). Most of these approaches are not suitable for workforce scheduling in retail with it's particular requirements. However, applications like service centre workforce scheduling [24] have similarities with the problem presented here [5], as workforce requirements are dynamic fluctuating, shifts can change in their length and beginning, over- and understaffing are possible and should be minimized while the number of possible shifts can become very large.

Systems that are currently in practical use often base on the constraint-satisfaction approach. An example was developed with ILOG tools [10]. Due to deregulation the workforce scheduling problem in the retail sector has nowadays a lot of soft constraints and nonlinearities, what makes it hard to compute practical solutions with classical optimisation techniques. To our knowledge this problem has not attracted a lot of researchers in retail for a long time [5]. The situation currently seems to change as the references [12], [14] and [17] were all published recently. Additionally, one can find related work on scheduling of employees with different skills, e.g. [24], [6] or approaches that take different employees types and their job satisfaction into account [15]. Prüm [19] addresses different variants of a personnel planning problem from retail, which is similar to our problem. The results indicate that problems of realistic size with constraints can in general not be successfully solved with exact methods. Therefore, in the remainder of this paper we focus on heuristic approaches for our application.

4 Constructive Approach

Sauer and Schumann suggest a constructive approach as part of a commercial interactive scheduling tool specifically designed for retail. This system allows for demand-oriented assignment planning, with the intervals at least 15 minutes long. The planning length is one week of seven days. Longer periods must be planned week by week. The approach cannot consider more than one workstation or sub-daily workstation rotations. However, planning of breaks is possible and employees can be divided in a primary and secondary pool for each scheduling task. These pools can be constructed according to different skill levels or departments. The approach is basically a greedy heuristic that produces a solution quickly and allows for the integration of human expert knowledge. For more details see [21]. The main scheduling scheme is presented in the Algorithm 1.

Algorithm 1. Constructive heuristic

```
while days to schedule do
    Select most difficult day to schedule
    Calculate maximum interval
    Chose available staff for the selected day
    Rank possible staff candidates
    if candidate exists then
        Calculate assignment interval
        Assign employee
        Update employees and demand data
    else
        Reduce demand
    end if
end while
```

The schedule is generated as a set of assignments under the assumption that the available staff is rare and understaffing will be ineluctable. The idea is to use a bottleneck heuristic. Most important decisions are the selection of days and the appropriate personnel. Here heuristic knowledge is incorporated.

Selecting a day: A day is schedulable if it has an open demand. If there exist more than one day with an open demand, the difficulty of the days is computed which is the ratio of the open demand of that day and the open demand of the week. The most difficult day is chosen. Then the longest open demand interval of this day is calculated and the next assignment has to be placed in this interval.

Selecting an employee for a day: Based on the information above the available staff can be identified. These are the employees that are not scheduled already on this day and are available for at least parts of the assignment interval. The candidates have to be evaluated according to their adequacy for the next assignment, computed by a number of factors, namely their normal weekly working time, time-sheet balance and a priority value of the employee which depends on the status group and working pool. This evaluation results in a total order of all possible candidates for the assignment. The highest rated employee will be assigned. During this selection constraints like the minimal assignment time for each employee are regarded.

5 Evolution Strategy

The evolution strategy (ES) is a well-known biologically-inspired metaheuristic [1]. Mutation is the main search operator employed. Our application is of a combinatorial nature, this requires some adaptation of the ES. Algorithm 2 presents an overview of the implemented ES.

Algorithm 2. Workforce-scheduling evolution strategy

```
Initialise the Population with $\mu$ Individuals
Repair the Population
Evaluate the $\mu$ Individuals
loop
    Recombination to generate $\lambda$ Offspring
    Mutate the $\lambda$ Offspring
    Repair the $\lambda$ Offspring
    Evaluate all repaired Individuals
    Selection $((\mu + \lambda)$ or $(\mu, \lambda))$
end loopUntil Criterion
```

To apply the ES, the problem needs to be conveniently represented. A two-dimensional matrix is applied. The rows represent the employees and the columns the time slots. The meaning of the matrix elements is as follows:

- 0: Store is closed or employee is absent (holiday, training, illness).
- 1: Employee is assigned to the workstation.
- 2: Employee is generally available but not dispatched in staffing.

The ES-population is initialized with randomized but valid solutions that make use of prior knowledge like the opening hours of the store, bank holidays and employee availability. Possible errors w.r.t. the requirements of working time models are repaired.

Ten alternative recombination variants were evaluated in a pre-test. The best performance was achieved with the classical one-point crossover: The same crossover point is determined independently and at random for all employees (row) of a solution and the associated parts of the two parents are exchanged.

Two different forms of mutation were devised. In standard-ES, mutation is performed using normally-distributed random variables so that small changes are more frequent than large ones. Mutation type 'N' of an offspring is carried out by picking an employee at random and changing the workstation assignment for a time interval chosen at random. The number of employees selected for mutation follows a $(0;\sigma)$-normal distribution. Results are rounded and converted to positive integer numbers. The mutation stepsize sigma is controlled self-adaptively using a log-normal distribution and intermediate recombination, following the standard scheme of ES [1].

The second approach for mutation (type 'E') has already proven to be efficient in a staff scheduling problem from logistics [9]. The concept of maximum entropy is used to select a specific mutation distribution from numerous potential candidates. This mutation is based on the work of Rudolph [20]. Main differences to Rudolph's approach are the introduction of dimension boundaries, the consideration of employee availability in mutation, and, finally, an increased mutation intensity due to the high-dimensional search space.

(μ,λ)-selection (comma-selection) as well as $(\mu+\lambda)$-selection (plus-selection) are used as well as different population sizes. The best solution found during an experimental run is always stored and updated in a "golden cage". It represents the final solution of the run. Following suggestions in the literature e.g. [1], the ratio μ/λ is set to 1/5 - 1/7 during the practical experiments. After mutation, a repair heuristic is applied to individuals to remove constraint violations in the following order, based on the observed error frequency (more details in [8]):

- Overstaffing: If possible, employees are reassigned to dummy workstation.
- Understaffing: If possible, additional employees are assigned to workstation.
- More than one working time model per day per employee: If possible, one time period is transferred to an employee, who has no assignment that day.
- Minimum length of a working time model: If possible, the interval is expanded, while shortening working hours of others.
- Maximum length of a working time model: If possible, the interval is split and one part is assigned to another employee.
- Correction of the working time models: All working time models are made consistent, regardless of effects concerning over- and understaffing.

6 Results and Discussion

The algorithms were tested on the retailer case study with the objective to minimise resulting error points under the given constraints. The implementation was done in C# on a 2.66 GHz quad core PC with 4 GB RAM under Windows Vista. Table 1 presents the results for different variants of the heuristics. The runs using ES were repeated 30 times for each parameter set. The constructive method requires approx. 2 minutes and computes a deterministic result. The ES (including repair) requires roughly 6 hours for a single run with 400,000

Table 1. Results with various parameter settings (averaged over 30 runs for ES)

heuristic	mean error	minimal error	standard deviation	understaffing in minutes	overstaffing in minutes demand > 0	too much weekly working time in minutes
constructive method (commercial) [21]	27660.0	27660	0.0	2280.0	2160.0	23220.0
ES(1,5) type N	**4910.0**	**3840**	425.7	2178.0	0.0	2881.2
ES(1+5) type N	8828.8	4320	2786.0	3496.0	2.0	5330.8
ES(10,50) type N	5996.4	4836	656.2	2616.0	0.0	3380.4
ES(10+50) type N	17440.4	6132	11405.3	4328.0	4.0	13108.4
ES(30,200) type N	6712.8	4680	1050.9	2786.0	0.0	3926.8
ES(30+200) type N	13219.6	5700	5829.6	3924.0	0.0	9295.6
ES(1,5) type E	8257.6	5316	2416.9	3272.0	6.0	4979.6
ES(1+5) type E	9809.6	4980	2827.0	3600.0	0.0	6209.6
ES(10,50) type E	8174.8	5136	1464.1	3278.0	0.0	4896.8
ES(10+50) type E	15316.8	7296	6638.5	4244.0	2.0	11070.8
ES(30,200) type E	7278.0	5040	1209.3	3118.0	2.0	4158.0
ES(30+200) type E	11503.6	6000	3608.0	3666.0	4.0	7833.6

fitness calculations. Both heuristics produce working time models that respect minimum and maximum length as set by the planner. In no case more than one working time model per day was assigned to any one employee. Finally, both solution approaches avoided overstaffing when demand for personnel was zero.

The commercial constructive heuristic, which was specifically developed for the requirements of retailers with one workstation, produces a large number of error points for overcoverage within the schedule. The method has difficulties generating fitting working time models for the present problem. Employees are often scheduled for too long time periods without good alignment to demand. With regard to undercoverage, the approach yields good results. This is, however, accomplished by exceeding weekly target hours, thus creating associated error points. Shortening some of the working time models would reduce errors in overcoverage and violation of contractually set working times.

ES performs significantly better. The best mean results were achieved with ES(1,5) and mutation type N, which is based on the classical normally-distributed form of mutation for continuous parameter optimisation. Thus, recombination is not an important search operator for this application. The assignment plans generated with the ES(1,5) can hardly be improved upon, even with highly complex manual changes. For this reason, and because of the vast improvement over the specialized constructive approach, these plans can be regarded as very usable. Interestingly, the mutation type E, specially designed for integer search spaces, does not consistently outperform the type N. This is in contrast to previous results produced for a similar problem in logistics [9].

Generally, the comma selection performs better than plus selection (for the same μ and λ). The comma strategy "forgets" the parent values after each generation, which allows for a temporary deterioration of objective function values. This is helpful in escaping from a local optimum as is also confirmed by the standard deviations. Plus selection always has a greater standard deviation than comma selection with the same μ and λ. With regard to improving solutions, a tendency can be seen in the comma strategy with classical mutation type

N toward smaller populations. Because of the uniform termination criterion of 400,000 fitness calculations, a smaller population means more iteration cycles. Many steps are required to arrive at a good plan. Thus, it seems preferable to track changes for more iterations as compared to richer diversity (through larger populations) of the solution space in each iteration. The effect is not so clearly visible for the ES with mutation type E which will need further investigation.

7 Conclusions

We focus on the problem of simultaneously assigning staff to workstations and generating optimised working time models on the basis of given demand. For a complex real-world application from retail, it was shown that the evolution strategy in a hybrid form which integrates a repair heuristic significantly outperforms a recent commercial constructive heuristic. Thus, hybrid metaheuristics, and ES in particular, are capable of finding better solutions to staff planning problems than heuristics that have been tailored to the domain. As the statistical spread of the ES-solutions and the results with different strategy parameter settings demonstrate, this outcome is rather stable. Stability of algorithms is important for practical problems as was pointed out in [22].

The computational effort of the ES is much higher than for the constructive method. This is acceptable here since workforce planning is not time-critical. Additionally, if the planning horizon was shortend, the computational requirements would be reduced accordingly. Particularly the ES-solutions generated with comma selection and small populations appear excellent. Repairing the violation of soft constraints significantly improved the quality of results for the ES. However, the repair comes at additional computational expense. These results are in line with our findings for a related scheduling problem from logistics [9] and for survivable network design [16]. Thus, we conclude that the ES, despite its original focus on continuous parameter optimisation, is a powerful tool in combinatorial optimisation and deserves more attention. However, a proper adaptation in terms of representation and operators are called for, possibly complemented by the integration of a repair heuristic. To broaden the basis for our conclusions, additional combinatorial applications will be investigated. Future research also looks at synergies between both planning approaches. For instance, the constructive heuristic could be randomized and used to generate initial solutions for the metaheuristic to reduce computational effort.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: A comprehensive introduction. Nat. Comp. 1, 3–52 (2002)
2. Castillo, I., Joro, T., Li, Y.Y.: Workforce scheduling with multiple objectives. EJOR 196, 162–170 (2009)
3. Cheang, B., Li, H., Lim, A., Rodrigues, B.: Nurse rostering problems – a bibliographic survey. EJOR 151(3), 447–460 (2003)

4. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D.: An annotated biblography of personnel scheduling and rostering. Annals of OR 127, 21–144 (2002)
5. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. EJOR 153(1), 3–27 (2004)
6. Fowler, J.W., Wirojanagud, P., Gel, E.S.: Heuristics for workforce planning with worker differences. EJOR 190(3), 724–740 (2008)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
8. Günther, M.: Hochflexibles Workforce Management. PhD dissertation. Ilmedia (2011) (to appear)
9. Günther, M., Nissen, V.: Sub-daily staff scheduling for a logistics service provider. KI 24(2), 105–113 (2010)
10. Hare, D.R.: Staff scheduling with ilog solver. Technical report, Okanagan University College (2007)
11. TU Ilmenau. Test data sub-daily staff scheduling (2010), `http://www.tu-ilmenau.de/wid/forschung/testprobleme-personaleinsatzplanung`
12. Kabak, Ö., Ülengin, F., Akta, E., Önsel, S., Topcu, Y.I.: Efficient shift scheduling in the retail sector through two-stage optimization. EJOR 184(1), 76–90 (2008)
13. Lesaint, D., Voudouris, C., Azarmi, N., Alletson, I., Laithwaite, B.: Field workforce scheduling. BT Technology Journal 21(4), 23–26 (2003)
14. Melachrinoudis, E., Min, H.: The hybrid queuing and bi-objective integer programming model for scheduling frontline employees in a retail organisation. Int. J. of Services Techn. and Management 9(1), 33–50 (2008)
15. Mohan, S.: Scheduling part-time personnel with availability restrictions and preferences to maximize employee satisfaction. Math. and Comp. Model. 48(11), 1806–1813 (2008)
16. Nissen, V., Gold, S.: Survivable network design with an evolution strategy. In: Yang, A., Shan, Y., Bui, L.T. (eds.) Success in Evolutionary Computation. SCI, pp. 263–283. Springer, Heidelberg (2008)
17. Pastor, R., Olivella, J.: Selecting and adapting weekly work schedules with working time accounts. EJOR 184(1), 1–12 (2008)
18. Pinedo, M.: Planning and Scheduling in Manufacturing and Service. Springer Series in Operations Research. Springer, New York (2005)
19. Prüm, H.: Entwicklung von Algorithmen zur Personaleinsatzplanung mittels ganzzahliger linearer Optimierung. Master thesis. FH Trier (2006)
20. Rudolph, G.: An evolutionary algorithm for integer programming. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 139–148. Springer, Heidelberg (1994)
21. Sauer, J., Schumann, R.: Modelling and solving workforce scheduling problems. In: Sauer, J., Edelkamp, S., Schattenberg, B. (eds.) 21. Workshop PUK, Osnabrück, pp. 93–101 (2007)
22. Schumann, R., Sauer, J.: Implications and consequences of mass customization on manufacturing control. In: Blecker, T., Edwards, K., Friedrich, G., Salvador, F. (eds.) Innovative Processes and Products for Mass Customization, Hamburg, GITO, pp. 365–378 (2007)
23. Tien, J., Kamiyama, A.: On manpower scheduling algorithms. SIAM Rev. 24(3), 275–287 (1982)
24. Valls, V., Perez, A., Quintanilla, S.: Skilled workforce scheduling in service centres. EJOR 193(3), 791–804 (2009)

Optimization of the Nested Monte-Carlo Algorithm on the Traveling Salesman Problem with Time Windows

Arpad Rimmel[1], Fabien Teytaud[2], and Tristan Cazenave[1]

[1] LAMSADE, Université Paris Dauphine, France
[2] TAO (Inria), LRI, Univ. Paris-Sud, France

Abstract. The traveling salesman problem with time windows is known to be a really difficult benchmark for optimization algorithms. In this paper, we are interested in the minimization of the travel cost. To solve this problem, we propose to use the nested Monte-Carlo algorithm combined with a Self-Adaptation Evolution Strategy. We compare the efficiency of several fitness functions. We show that with our technique we can reach the state of the art solutions for a lot of problems in a short period of time.

Keywords: Nested Monte-Carlo, Self Adaptation, Traveling Salesman, Time Window.

1 Introduction

The traveling salesman problem is a difficult optimization problem and is used as a benchmark for several optimization algorithms. In this paper we tackle the problem of optimizing the Traveling Salesman Problem with Time Windows (TSPTW). For solving this problem, we combine a nested Monte-Carlo algorithm [4] and an evolutionary algorithm. With this system, as we will see, the important point is that we will have to optimize a function which is noisy and where the evaluation is not the score on average but the best score among a certain number of runs. When the noise is uniform on the whole function, optimizing for the mean or for the min is equivalent, so we will focus on problems where the noise is non uniform. We will show on an artificial problem that having a fitness function during the optimization that is different from the one we want to optimize can improve the convergence rate. We will then use this principle to optimize the parameters of a nested Monte-Carlo algorithm for the TSPTW.

We have chosen the use of Evolution-Strategies (ES [12]) for the optimization part. This kind of algorithms are known to be simple and robust. See [12,2] for more details on ES in general.

The paper is organized as follows : Section 2 presents the optimization algorithm used, Section 3 is the presentation of the artificial problem and the results we obtain, Section 4 is the application to the TSPTW, and finally we discuss all the results in the Section 5.

© Springer-Verlag Berlin Heidelberg 2011

2 Self-Adaptation Evolution Strategy

In all the paper we use $(\mu/\mu, \lambda)$-ES. λ is the population size, μ the number of parents (the selected population size), and the parents are selected only among individuals belonging to the new generated population (and not in the mixing between the new generated population and the previous parents). Mutation will be done according to a Gaussian distribution.

We have chosen the Self-Adaptation Evolution Strategy (SA-ES) for the optimization of our problem. This algorithm has been introduced in [12] and [14]. An extended version with full covariance matrix has been proposed in [3]. An improvement of this algorithm, based on the selection ratio, efficient in the case of large population can also be used [16]. In our experiments, we use the standard SA-ES with small population sizes, then we do not use this last improvement. The motivation behind the choice of this algorithm is that it is known to be really robust, because it doesn't need any a priori knowledge on the problem. The SA-ES algorithm is presented in Algorithm 1.

Algorithm 1. Mutative self-adaptation algorithm

```
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; $f_{(1)} < f_{(2)} < \cdots < f_{(\lambda)}$.
  $z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z_{(i)}$
  $\sigma^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} \sigma_{(i)}$
  $y = y + z^{avg}$
end while
```

3 Noisy Sphere

We will first optimize on an artificial problem: the noisy sphere.

3.1 Presentation

The noisy sphere is a classical artificial problem for optimization experiments. However, here, the noise function will be original. The noise will be Gaussian and non uniform. We will use 5 dimensions.

The exact equation of the function that we will use is the following:

$$f(y) = \sum_{i=1}^{N} (y_i^2 + N(0, (2y_i)^2)).$$

It is represented on the top-left of figure 1.

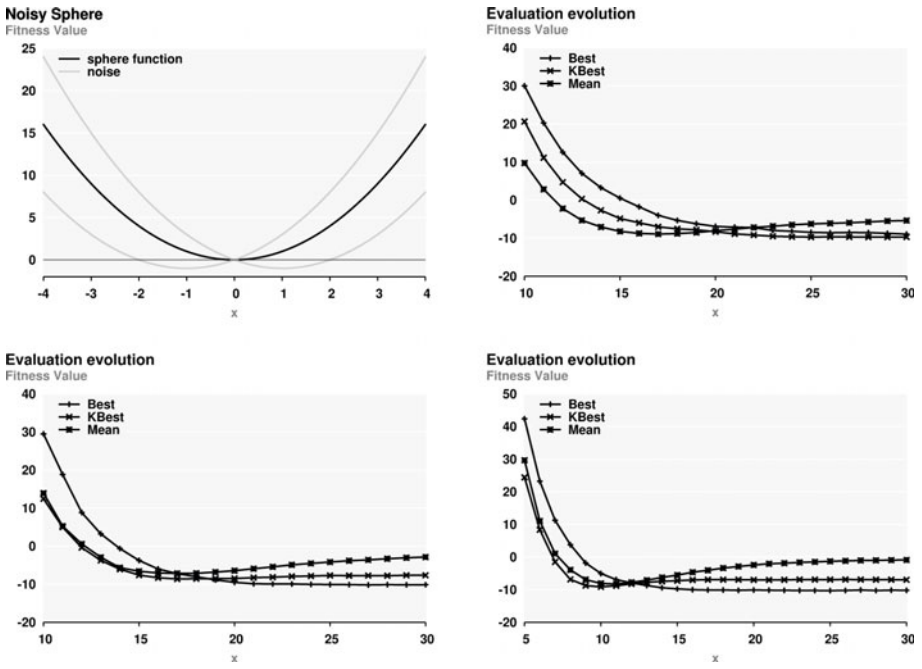


Fig. 1. Top-left. Representation of the evaluation function. **Top-right, bottom-left, bottom-right.** Evolution of the true score as a function of the iterations with $n = 10$, $n = 100$ and $n = 300$ respectively.

The evaluation function $eval(f(y))$ will be the min over 1000 runs of f with parameters y. We will optimize the expectation of this function:

$$eval(f(y)) = \min(f^i(y), i = 1..1000),$$

f^i being the i-th run of f.

During the optimization, we will use a fitness function to evaluate an individual. Usually, people use the same function for the fitness function and for the evaluation function. However, we see that the function that we want to optimize is very unstable. For this reason, we propose to use a fitness function that can be different from the evaluation function.

The 3 fitness functions that we will use are:

- $best_n(f(y)) = \min(f^i(y), i = 1..n)$
- $mean_n(f(y)) = \frac{\sum_{i=1}^{n} f^i(y)}{n}$
- $kbest_{k,n}(f(y)) = \frac{\sum_{i=1}^{k} f^i(y)}{n}$ with $f^1(y) < f^2(y) < ... < f^k(y) < ... < f^n(y)$

As the fitness function used during the optimization is not the same as the evaluation function, we compute for each generation the score of the best individual according to the evaluation function. This function is noisy, so the **true score** of an individual will be the average over $NbEval$ runs of the evaluation function.

This is very costly in number of evaluations but this true score is only used to show that the algorithm converges and will only be used for the noisy sphere.

3.2 Experiments

We use the SA algorithm to do the optimizations. In the experiments, we used $k = 5$ for *kbest* and $NbEval = 100$.

We compare $best_n$, $mean_n$ and $kbest_{5,n}$ for different values of n.

We compute the true score of the best individual in function of the number of the generation. Every curve is the average over 30 runs.

The results are given on the figure 1.

We see that in every cases, the convergence is slower with *best* than with *mean* and *kbest*. However, the final value is always better for *best*. This is because *best* is the fitness function the most similar to the evaluation function.

For high values of n, the convergence is equivalent for *kbest* and *mean*. Furthermore, the final value is better for *kbest* than for *mean*. This implies that for high value of n, it is always better to use *kbest* instead of *mean*.

For small values of n, *kbest* converges slowly than *mean* but achieves a better final value.

As a conclusion, the choice of the fitness function will depend on the need of the user. If the speed of the convergence is important, one can use *mean* or *kbest* depending on n. If the final value is important, *best* is the function to use.

We will now see if the conclusions we obtained on an artificial problem are still valid when we optimize on difficult benchmarks.

4 Application

We will now focus on the TSPTW problem. First, we will describe the problem. Then, we will present the nested Monte-Carlo algorithm. Finally, we will show the results we obtain when we optimize the parameters of the algorithm on TSPTW.

4.1 Traveling Salesman Problem with Time Windows

The traveling salesman problem is an important logistic problem. It is used to represent the problem of finding an efficient route to visit a certain number of customers, starting and finishing at a depot. The version with time windows adds the difficulty that each customer has to be visited within a given period of time. The goal is to minimize the length of the travel. TSPTW is an NP-hard problem and even finding a feasible solution is NP-complete [13]. Early works [5,1] were based on branch-and-bound. Later, Dumas et al. used a method based on Dynamic programming [6]. More recently, methods based on constraints programming have been proposed [10,7].

Algorithms based on heuristics have also been considered [15,8].

Finally, [9] provides a comprehensive survey of the most efficient methods to solve the TSPTW and proposes a new algorithm based on ant colonies that achieves very good results. They provide a clear environment to compare algorithms on a set of problems that we used in this article.

Technical description of the Traveling Salesman Problem with Time Windows. The TSP can be described as follow. Let G be an undirected complete graph. $G = (N, A)$ where $N = 0, 1, ..., n$ is a set of nodes and $A = N * N$ is the set of edges between the nodes. The node 0 represents the depot. The n other nodes represent the customers. A cost function $c : A \rightarrow \mathbb{R}$ is given. It represents the distance between 2 customers. A solution to this problem is a sequence of nodes $P = (p_0, p_1, ..., p_n, p_{n+1})$ where $p_0 = p_{n+1} = 0$ and $(p_1, ..., p_n)$ is a permutation of $N \setminus \{0\}$.

The goal is to minimize the function

$$cost(P) = \sum_{k=0}^{n} c(a_{p_k,p_{k+1}}).$$

In the version with time windows, each customer i is associated with an interval $[e_i, l_i]$. The customer must not be served before e_i or after l_i. It is allowed to arrive at a node i before e_i but the departure time becomes e_i.

Let d_{p_k} be the departure time from node p_k, $d_{p_k} = \max(r_{p_k}, e_{p_k})$ where r_{p_k} is the arrival time at node p_k.

The function to minimize is the same but a set of constraints must now be respected. Let $\Omega(P)$ be the number of windows constraints violated by tour P. The optimization of f must be done while respecting the following equation

$$\Omega(P) = \sum_{k=0}^{n+1} \omega(p_k) = 0,$$

where

$$\omega(p_k) = \begin{cases} 1 \text{ if } r_{p_k} > l_{p_k} \\ 0 \text{ otherwise} \end{cases},$$

and

$$r_{p_{k+1}} = \max(r_{p_k}, e_{p_k}) + c(a_{p_k,p_{k+1}}).$$

With the addition of the constraints, the problem becomes much more complicated and classical algorithms used for TSP are not efficient anymore. That is why we will use Nested Monte-Carlo which is described in the next part of the article.

4.2 Adaptation of the Nested Monte-Carlo Algorithm for the Traveling Salesman Problem with Time Windows

The Nested Monte-Carlo (NMC) algorithm [4] is a tree search algorithm. The tree is supposed to be large and the leaves of the tree (final positions of the problem) can be evaluated. It does not require any knowledge on the problem and is quite simple to implement. It is particularly efficient on problems where later decisions are as important as early ones. NMC has allowed to establish world records in single-player games such as Morpion Solitaire or SameGame. We first describe the NMC algorithm and then explain how we introduced heuristics in order to obtain better results on the TSPTW problem.

The Nested Monte-Carlo Algorithm. The NMC algorithm uses several levels. Each level uses the lower level to determine which action will be selected at each step. The level 0 is a Monte-Carlo simulation, i.e. a random selection of actions until a final position is reached. More precisely, in each position, a NMC search of level n will perform a level $n-1$ NMC for each action and then select the one with the best score. For example, a NMC search of level 1 will do a Monte-Carlo simulation for each action (those reaching a final position which can be evaluated) and select the action associated with the highest evaluation. Once an action has been selected, the problem is in a new position and the selection method is repeated again until a final position is reached. The performance of the algorithm is greatly improved by memorizing the best sequence for each level.

Algorithm 2. Nested Monte-Carlo

```
nested($position, level$)
best playout ← {}
while not end of the game do
  if $level = 1$ then
    $move \leftarrow \arg\max_m$(MonteCarlo(play($position, m$)))
  else
    $move \leftarrow \arg\max_m$(nested(play($position, m$), $level - 1$))
  end if
  if score of playout after $move$ > score of the best playout then
    best playout ← playout after $move$
  end if
  $position \leftarrow$ play($position$, move of the best playout)
end while
return score($position$)
```

play($position, m$) is a function that returns the new position obtained after having selected the action m in *position*

MonteCarlo($position$) is a function that returns the evaluation of the final position reached after having selected random actions from *position*.

NMC provides a good compromise between exploration and exploitation. It is particularly efficient for one-player games and gives good results even without domain knowledge. However, the results can be improved by the addition of heuristics.

Adaptation for the Traveling Salesman Problem with Time Windows. It is possible to improve the performance of NMC by modifying the Monte-Carlo simulations. An efficient way is to select actions based on heuristics instead of a uniform distribution. However, some randomness must be kept in order to preserve the diversity of the simulations.

To do that, we use a Boltzmann softmax policy. This policy is defined by the probability $\pi_\theta(p, a)$ of choosing the action a in a position p:

$$\pi_\theta(p, a) = \frac{e^{\phi(p,a)^T\theta}}{\sum_b e^{\phi(p,b)^T\theta}},$$

where $\phi(p, a)$ is a vector of features and θ is a vector of feature weights.

The features we use are the heuristics described in [15]:

- the distance to the last node: $h1(p,a) = c(d,a)$
- the amount of time that will be necessary to wait if a is selected because of the beginning of its time window: $h2(p,a) = \max(0, e_a - (T_p + c(d,a)))$
- the amount of time left until the end of the time window of a if a is selected: $h3(p,a) = \max(0, l_a - (T_p + c(d,a)))$

where d is the last node selected in position p, T_p is the time used to arrive in situation p, e_a is the beginning of the time window for action a, l_a is the end of the time window for the action a and $c(d,a)$ is the travel cost between d and a.

The values of the heuristic are normalized before being used.

The values that we will optimize are the values from the vector θ (the feature weights).

4.3 Experiments

We use the set of problems given in [11].

As we have 3 different heuristics, the dimension of the optimization problem is 3.

We define $NMC(y)$, the function that associates a set of parameters y to the permutation obtained by a run of the NMC algorithm, with parameters y on a particular problem.

The score $Tcost(p)$ of a permutation p is the travel cost. However, as the NMC algorithm can generate permutations with some windows constraints not respected, we added a constant to the score for each one.

$$Tcost(p) = cost(p) + 10^6 * \Omega(p),$$

$cost(p)$ is the cost of the travel p and $\Omega(p)$ the number of non-respected constraints.

10^6 is a constant high enough for the algorithm to first optimize $\Omega(p)$ and then $cost(p)$.

The exact equation of the function f that we will use is the following:

$$f(y) = Tcost(NMC(y)).$$

As the end of the evaluation, we want to obtain a NMC algorithm that we will launch for a longer period of time in order to obtain one good score on a problem. So the evaluation function should be the min on this period of time. As this period of time is not known and a large period of time would be too time-consuming, we arbitrarily choose a time of 1s to estimate the true score of an individual.

The evaluation function $eval(f(y))$ will be the min over r runs of f with parameters y. r being the amount of runs that can be done in 1s. It means that we want to optimize the expectation of this function:

$$eval(f(y)) = \min_{1s}(f(y)).$$

As for the sphere problem, we will use 3 different fitness functions instead of the evaluation function: $mean_n$, $kbest_n$ and $best_n$. In the experiments, we use $n = 100$.

We use a nested algorithm of level 2.

Optimization on one Problem. The first experiments are done on the problem rc206.3 which contains 25 nodes.

In this experiment we compare $best_{100}$, $kbest_{100}$ and $mean_{100}$. As in all the paper, the population size λ is equal to 12 and the selected population size μ is 3, and $\sigma = 1$. The initial parameters are $[1, 1, 1]$ and the stopping criterion of the evolution-strategy is 15 iterations. Results are the average of three independent runs.

Table 1. Evolution of the true score on the problem rc206.3

Iterations	BEST	KBEST	MEAN
1	2.7574e+06	2.4007e+06	2.3674e+06
2	5.7322e+04	3.8398e+05	1.9397e+05
3	7.2796e004	1.6397e+05	618.22
4	5.7274e+04	612.60	606.68
5	2.4393e+05	601.15	604.10
6	598.76	596.02	602.96
7	599.65	596.19	603.69
8	598.26	594.81	600.79
9	596.98	591.64	602.54
10	595.13	590.30	600.14
11	590.62	591.38	600.68
12	593.43	589.87	599.63
13	594.88	590.47	599.24
14	590.60	589.54	597.58
15	589.07	590.07	599.73

There is a lot of difference between the initial parameters and optimized parameters in term of performances. This shows that optimizing the parameters is really important in order to obtain good performance.

Results are similar as in the case of our noisy sphere function. $best_{100}$ reaches the best score, but converges slowly. $mean_{100}$ has the fastest convergence, but finds the worst final score. As expected, $kbest_{100}$ is a compromise between the two previous fitness, with a nice convergence speed and is able to find a score really close to the best. For this reason, we have chosen to use *kbest* for the other problems.

Results on all the problems. We launched the optimization algorithm on all the problems from the set in the paper from Potvin and Bengio [11]. We compare the best score we obtained on each problem with our algorithm and the current best known score from the literature. The results are presented in table 2. We provide the Relative Percentage Deviation (RPD): $100 * (value - bestknown)/bestknown$.

Table 2. Results on all problems from the set from Potvin and Bengio [11]. First Row is the problem, second column the number of nodes, third column the best score found in [9], forth column the best score found by algorithm and fifth column it the RPD. The problems where we find the best solutions are in bold. We can see that for almost all problems with our simple algorithm we can find the best score.

Problem	n	State of the art	Our best score	RPD	Problem	n	State of the art	Our best score	RPD
rc201.1	20	444.54	**444.54**	0	rc205.1	14	343.21	**343.21**	0
rc201.2	26	711.54	**711.54**	0	rc205.2	27	755.93	**755.93**	0
rc201.3	32	790.61	**790.61**	0	rc205.3	35	825.06	828.27	0.39
rc201.4	26	793.64	**793.64**	0	rc205.4	28	760.47	**760.47**	0
rc202.1	33	771.78	776.47	0.61	rc206.1	4	117.85	**117.85**	0
rc202.2	14	304.14	**304.14**	0	rc206.2	37	828.06	839.18	1.34
rc202.3	29	837.72	**837.72**	0	rc206.3	25	574.42	**574.42**	0
rc202.4	28	793.03	**793.03**	0	rc206.4	38	831.67	859.07	3.29
rc203.1	19	453.48	**453.48**	0	rc207.1	34	732.68	743.29	1.45
rc203.2	33	784.16	**784.16**	0	rc207.2	31	701.25	707.74	0.93
rc203.3	37	817.53	837.72	2.47	rc207.3	33	682.40	687.58	0.76
rc203.4	15	314.29	**314.29**	0	rc207.4	6	119.64	**119.64**	0
rc204.1	46	868.76	899.79	3.57	rc208.1	38	789.25	797.89	1.09
rc204.2	33	662.16	675.33	1.99	rc208.2	29	533.78	536.04	0.42
rc204.3	24	455.03	**455.03**	0	rc208.3	36	634.44	641.17	1.06

There is a lot of differences between one set of parameters optimized on one problem and one set of parameters optimized on an other problem. So, the optimization has to be done on each problem.

We obtain as well as the state of the art for all the problems with less than 29 nodes. We find at least one correct solution for each problem. When the number of nodes increases, this is not a trivial task. For problems more difficult with a higher number of nodes, we don't do as well as the best score. However, we still manage to find a solution close to the current best one and did this with little domain knowledge.

5 Conclusion

In this paper we used a new method for solving the TSPTW problem based on the optimization of a nested Monte-Carlo algorithm with SA. This algorithm is a generic algorithm, used in many different applications. The only adaptation to the TSPTW was to add 3 heuristics. Even in this case, for all the problems with less than 29 nodes, we were able to reach state of the art solutions with small computation times. However, a clear limitation of our algorithm is dealing with a large number of nodes. A solution could be to prune some moves at the higher level of NMC. Other further work will be to add new heuristics. In this case, because of the higher dimensionality, we will try other evolution algorithms and increase the population size. A natural choice is the Covariance Matrix Self-Adaptation [3], known to be robust and good for large population sizes. Adding covariance and allowing large population sizes should increase the speed of the convergence.

References

1. Baker, E.: An exact algorithm for the time-constrained traveling salesman problem. Operations Research 31(5), 938–945 (1983)
2. Beyer, H.-G.: The Theory of Evolution Strategies. Natural Computing Series. Springer, Heidelberg (2001)
3. Beyer, H.-G., Sendhoff, B.: Covariance matrix adaptation revisited - the CMSA evolution strategy. In: Rudolph, G., Jansen, T., Lucas, S.M., Poloni, C., Beume, N. (eds.) Proceedings of PPSN, pp. 123–132 (2008)
4. Cazenave, T.: Nested Monte-Carlo search. In: IJCAI, pp. 456–461 (2009)
5. Christofides, N., Mingozzi, A., Toth, P.: State-space relaxation procedures for the computation of bounds to routing problems. Networks 11(2), 145–164 (1981)
6. Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M.: An optimal algorithm for the traveling salesman problem with time windows. Operations Research 43(2), 367–371 (1995)
7. Focacci, F., Lodi, A., Milano, M.: A hybrid exact algorithm for the TSPTW. INFORMS Journal on Computing 14(4), 403–417 (2002)
8. Gendreau, M., Hertz, A., Laporte, G., Stan, M.: A generalized insertion heuristic for the traveling salesman problem with time windows. Operations Research 46(3), 330–335 (1998)
9. López-Ibáñez, M., Blum, C.: Beam-ACO for the travelling salesman problem with time windows. Computers & OR 37(9), 1570–1583 (2010)
10. Pesant, G., Gendreau, M., Potvin, J., Rousseau, J.: An exact constraint logic programming algorithm for the traveling salesman problem with time windows. Transportation Science 32(1), 12–29 (1998)
11. Potvin, J., Bengio, S.: The vehicle routing problem with time windows part II: genetic search. INFORMS Journal on Computing 8(2), 165 (1996)
12. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
13. Savelsbergh, M.: Local search in routing problems with time windows. Annals of Operations Research 4(1), 285–305 (1985)
14. 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, Juli (1974)
15. Solomon, M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35(2), 254–265 (1987)
16. Teytaud, F.: A new selection ratio for large population sizes. In: Applications of Evolutionary Computation, pp. 452–460 (2010)

Author Index