Lecture Notes in Computer Science 7835

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

Anna I. Esparcia-Alcázar et al. (Eds.)

Applications of Evolutionary Computation

16th European Conference, EvoApplications 2013
Vienna, Austria, April 3-5, 2013
Proceedings

Springer

Volume Editors

see next page

Front cover EvoStar 2013 logo by Kevin Sim, Edinburgh Napier University

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-37191-2 e-ISBN 978-3-642-37192-9
DOI 10.1007/978-3-642-37192-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013933106

CR Subject Classification (1998): F.2, I.2.6, I.2.8-9, G.1.6, I.5, C.2, F.1, K.4.4, J.1, J.2, J.7, K.8.0

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

© Springer-Verlag Berlin Heidelberg 2013

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 ist 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

Anna I Esparcia-Alcázar
S2 Grupo, Spain
aesparcia@s2grupo.es

Sara Silva
INESC-ID Lisboa, Portugal
sara@kdbio.inesc-id.pt

Alexandros Agapitos
University College Dublin, Ireland
alexandros.agapitos@ucd.ie

Carlos Cotta
Universidad de Málaga, Spain
ccottap@lcc.uma.es

Ivanoe De Falco
ICAR–CNR, Italy
ivanoe.defalco@na.icar.cnr.it

Antonio Della Cioppa
University of Salerno, Italy
adellacioppa@unisa.it

Konrad Diwold
Fraunhofer IWES, Germany
konrad.diwold@iwes.fraunhofer.de

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

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

Francisco Fernández de Vega
Universidad de Extremadura, Spain
fcofdez@unex.es

Paolo Burelli
Aalborg University Copenhagen
Denmark
pabu@create.aau.dk

Kevin Sim
Edinburgh Napier University, UK
K.Sim@napier.ac.uk

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

Anabela Simões
Coimbra Polytechnic, Portugal
abs@isec.pt

JJ Merelo
Universidad de Granada, Spain
jmerelo@geneura.ugr.es

Neil Urquhart
Edinburgh Napier University, UK
n.urquhart@napier.ac.uk

Evert Haasdijk
VU University Amsterdam
The Netherlands
e.haasdijk@vu.nl

Mengjie Zhang
Victoria University of Wellington
New Zealand
mengjie.zhang@ecs.vuw.ac.nz

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

A.E. Eiben
VU University Amsterdam
The Netherlands
a.e.eiben@vu.nl

Andrea Tettamanzi
Université de Nice Sophia Antipolis, France
andrea.tettamanzi@unice.fr

Kyrre Glette
University of Oslo, Norway
kyrrehg@ifi.uio.no

Philipp Rohlfshagen
SolveIT Software, Australia
philipp.r@gmail.com

Robert Schaefer
AGH University of Science
and Technology, Poland
schaefer@agh.edu.pl

Preface

Evolutionary computation (EC) techniques are efficient, nature-inspired planning and optimization methods based on the principles of natural evolution and genetics. Owing to their efficiency and simple underlying principles, these methods can be used in the context of problem solving, optimization, and machine learning. A large and continuously increasing number of researchers and professionals make use of EC techniques in various application domains. This volume presents a careful selection of relevant EC examples combined with a thorough examination of the techniques used in EC. The papers in the volume illustrate the current state of the art in the application of EC and should help and inspire researchers and professionals to develop efficient EC methods for design and problem solving.

All papers in this book were presented during EvoApplications 2013, which incorporates a range of tracks on application-oriented aspects of EC. Originally established as EvoWorkshops in 1998, it provides a unique opportunity for EC researchers to meet and discuss application aspects of EC and has been an important link between EC research and its application in a variety of domains. During these 15 years, new workshops and tracks have arisen, some have disappeared, while others have matured to become conferences of their own, such as EuroGP in 2000, EvoCOP in 2004, EvoBIO in 2007, and EvoMUSART only last year.

EvoApplications is part of EVO*, Europe's premier co-located events in the field of evolutionary computing. EVO* was held in Vienna, Austria, during April 3–5, 2013, and 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; and EvoMUSART, the main International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design. The proceedings for all of these events in their 2013 edition are also available in the LNCS series (volumes 7831, 7832, 7833, and 7834, respectively).

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

EvoApplications 2013 consisted of the following individual tracks:

- *EvoCOMNET*, track on nature-inspired techniques for telecommunication networks and other parallel and distributed systems
- *EvoCOMPLEX*, track on evolutionary algorithms and complex systems
- *EvoENERGY*, track on EC in energy applications
- *EvoFIN*, track on evolutionary and natural computation in finance and economics
- *EvoGAMES*, track on bio-inspired algorithms in games
- *EvoIASP*, track on EC in image analysis signal processing and pattern recognition
- *EvoINDUSTRY*, track on nature-inspired techniques in industrial settings
- *EvoNUM*, track on bio-inspired algorithms for continuous parameter optimization
- *EvoPAR*, track on parallel implementation of evolutionary algorithms
- *EvoRISK*, track on computational intelligence for risk management, security and defence applications
- *EvoROBOT*, track on EC in robotics
- *EvoSTOC*, track on evolutionary algorithms in stochastic and dynamic environments

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 interorganizational networks, etc. To address the challenges of these systems, this track promotes the study and the application of strategies inspired by the observation of biological and evolutionary processes, that 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 in 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.

EvoIASP, the longest-running of all EvoApplications tracks that celebrated its 15th edition this year, was 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.

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

EvoPAR covers all aspects of the application of parallel and distributed systems to EC as well as the application of evolutionary algorithms for improving parallel architectures and distributed computing infrastructures. EvoPAR focuses on the application and improvement of distributed infrastructures, such as grid and cloud computing, peer-to-peer (P2P) system, as well as parallel architectures, GPUs, manycores, etc. in cooperation with evolutionary algorithms.

EvoRISK focuses on challenging problems in risk management, security, and defence, and covers both theoretical developments and applications of computational intelligence to subjects such as cyber crime, IT security, resilient and self-healing systems, risk management, critical infrastructure protection (CIP), military, counter-terrorism and other defence-related aspects, disaster relief, and humanitarian logistics.

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, providing the first platform to present and discuss the latest research in this field.

In line with its tradition of adapting the list of the tracks to the needs and demands of the researchers working in the field of evolutionary computing, new tracks have arisen this year while others have been discontinued. This edition saw the birth of four new tracks: EvoENERGY, EvoINDUSTRY, EvoROBOT, and a General track, for papers dealing with applications not covered by any of the established tracks. EvoROBOT, however, is not completely new as it goes back to 15 years ago, already being present in the first edition of the EvoWorkshops.

The number of submissions to EvoApplications 2013 increased compared to the previous edition, reaching a total of 119 entries (compared to 90 in 2012 and 162 in 2011). Table 1 shows relevant submission/acceptance statistics, with the figures for the 2012 edition also reported.

Table 1. Submission/acceptance statistics for EvoApplications 2013 and 2012

	2013			2012		
	Submissions	Accept	Ratio	Submissions	Accept	Ratio
EvoCOMNET	12	8	67%	6	4	67%
EvoCOMPLEX	9	7	78%	13	9	69%
EvoENERGY	8	5	63%	-	-	-
EvoFIN	11	6	55%	9	6	67%
EvoGAMES	9	7	78%	13	9	69%
EvoIASP	28	12	43%	13	7	54%
EvoINDUSTRY	5	2	40%	-	-	-
EvoNUM	12	3	25%	12	4	33%
EvoPAR	5	4	80%	10	8	80%
EvoRISK	2	1	50%	2	1	50%
EvoROBOT	11	7	64%	-	-	-
EvoSTOC	6	3	50%	7	3	43%
General track	1	0	0%	-	-	-%
Total	119	65	55%	90	54	60%

As for previous years, accepted papers were split into oral presentations and posters, with the paper length for these two categories being the same for all the tracks. The low acceptance rate of 55% for EvoApplications 2013, along with the significant number of submissions, is an indicator of the high quality of the articles presented at the conference, showing the liveliness of the scientific movement in the corresponding fields.

Many people helped make EvoApplications a success. We would like to express our gratitude firstly to the authors for submitting their work, to the members of the Program Committees for devoting their energy to reviewing these papers, and to the audience for their lively participation.

We would also like to thank the Institute for Informatics and Digital Innovation at Edinburgh Napier University, UK, for their coordination efforts.

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 a conference of this kind, with a large number of different organizers and different opinions, would be unmanageable.

Further thanks to the local organizing team: Bin Hu, Doris Dicklberger, and Günther Raidl from the Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology, for making the organization of such an event possible in a place as unique as Vienna.

Last but surely not least, we want to especially acknowledge Şima Etaner-Uyar for her hard work as Publicity Chair of the event, Kevin Sim as webmaster, and Marc Schoenauer for his continuous help in setting up and maintaining the MyReview management software.

April 2013

Anna I. Esparcia-Alcázar
Sara Silva
Kevin Sim
Stefano Cagnoni
Alexandros Agapitos
Anabela Simões
Neil Urquhart
Carlos Cotta
Ivanoe De Falco
Evert Haasdijk
J.J. Merelo
Antonio Della Cioppa
Giovanni Squillero
Francisco Fernández de Vega
Ernesto Tarantino
Konrad Diwold
Mengjie Zhang
Anikó Ekárt
Andrea Tettamanzi
A.E. Eiben
Paolo Burelli
Philipp Rohlfshagen
Robert Schaefer
Kyrre Glette

Organization

EvoApplications 2013 was part of EVO* 2013, Europe's premier co-located events in the field of evolutionary computing, which also included the conferences EuroGP 2013, EvoCOP 2013, EvoBIO 2013, and EvoMUSART 2013

Organizing Committee

EvoApplications Chair:	Anna Isabel Esparcia-Alcázar, S2 Grupo, Spain
Local Chair:	Bin Hu, TU Wien, Austria
Publicity Chair:	A. Şima Etaner-Uyar, Istanbul Technical University, Turkey
Webmaster:	Kevin Sim, Edinburgh Napier University, Scotland, UK
EvoCOMNET Co-chairs:	Ivanoe De Falco, ICAR–CNR, Italy Antonio Della Cioppa, University of Salerno, Italy Ernesto Tarantino, ICAR–CNR, Italy
EvoCOMPLEX Co-chairs:	Carlos Cotta, Universidad de Málaga, Spain Robert Schaefer, AGH University of Science and Technology, Poland
EvoENERGY Co-chairs:	Konrad Diwold, Fraunhofer IWES, Germany Kyrre Glette, University of Oslo, Norway
EvoFIN Co-chairs:	Andrea Tettamanzi, Université de Nice Sophia Antipolis, France Alexandros Agapitos, University College Dublin, Ireland
EvoGAMES Co-chairs:	J.J. Merelo, Universidad de Granada, Spain Paolo Burelli, Aalborg University Copenhagen, Denmark
EvoIASP Co-chairs:	Stefano Cagnoni, University of Parma, Italy Mengjie Zhang, Victoria University of Wellington, New Zealand
EvoINDUSTRY Co-chairs:	Neil Urquhart, Edinburgh Napier University, Scotland, UK Kevin Sim, Edinburgh Napier University, Scotland, UK

EvoNUM Co-chairs:	Anna Isabel Esparcia-Alcázar, S2 Grupo, Spain Anikó Ekárt, Aston University, UK
EvoPAR Co-chairs:	J.J. Merelo, Universidad de Granada, Spain Francisco Fernández de Vega, Universidad de Extremadura, Spain
EvoRISK Co-chairs:	Anna Isabel Esparcia-Alcázar, S2 Grupo, Spain Sara Silva, INESC-ID, Portugal
EvoROBOT Co-chairs:	A.E. Eiben, VU University Amsterdam, The Netherlands Evert Haasdijk, VU University Amsterdam, The Netherlands
EvoSTOC Co-chairs:	Anabela Simões, Coimbra Polytechnic, Portugal Philipp Rohlfshagen, SolveIT Software, Australia
General Track Chair:	Giovanni Squillero, Politecnico di Torino, Italy

Program Committees

EvoCOMNET Program Committee:

Baris Atakan	Koc University, Turkey
Mehmet E. Aydin	University of Bedfordshire, UK
Alexandre Caminada	University of Technology Belfort-Montbéliard, France
Iacopo Carreras	CREATE-NET, Italy
Gianni Di Caro	IDSIA, Switzerland
Muddassar Farroq	FAST National University of Computer and Emerging Technologies, Pakistan
Bryant Julstrom	St. Cloud State University, USA
Farrukh A. Khan	National University of Computer and Emerging Sciences, Pakistan
Kenji Leibnitz	Osaka University, Japan
Domenico Maisto	ICAR-CNR, Italy
Roberto Montemanni	IDSIA, Switzerland
Enrico Natalizio	INRIA Lille, France
Conor Ryan	University of Limerick, Ireland
Chien-Chung Shen	University of Delaware, USA
Lidia Yamamoto	University of Strasbourg, France
Nur Zincir-Heywood	Dalhousie University, Canada

EvoCOMPLEX Program Committee:

Tiago Baptista	Universidade de Coimbra, Portugal
Marc Ebner	Ernst Moritz Arndt Universität Greifswald, Germany
Carlos Fernandes	Universidad de Granada, Spain
Antonio J. Fernández Leiva	Universidad de Málaga, Spain
José E. Gallardo	Universidad de Málaga, Spain
Carlos Gershenson	UNAM, Mexico
Juan L. Jiménez	Université du Luxembourg
Iwona Karcz-Dulęba	Wrocław University of Technology, Poland
Juan J. Merelo	Universidad de Granada, Spain
Joshua L. Payne	University of Zurich, Switzerland
Katya Rodríguez-Vázquez	UNAM, Mexico
Giovanni Squillero	Politecnico di Torino, Italy
Maciej Smołka	AGH University of Science and Technology, Poland
Marco Tomassini	Université de Lausanne, Switzerland
Alberto Tonda	Institut des Systèmes Complexes, France

EvoENERGY Program Committee:

Istvan Erlich	University of Duisburg-Essen, Germany
Nicola Hochstrate	Lübeck University of Applied Sciences, Germany
Paul Kaufmann	Fraunhofer IWES, Germany
Martin Middendorf	University of Leipzig, Germany
Julian F. Miller	University of York, England, UK
Maizura Mokhtar	University of Central Lancashire, England, UK
Frank Neumann	University of Adelaide, Australia
Peter Palensky	Austrian Institute of Technology, Austria
Jan Ringelstein	Fraunhofer IWES, Germany
Gareth A. Taylor	Brunel University, England, UK
Andy Tyrrell	University of York, England, UK

EvoFIN Program Committee:

Anthony Brabazon	University College Dublin, Ireland
Dietmar Maringer	University of Basel, Switzerland
Michael O'Neill	University College Dublin, Ireland
David Edelman	University College Dublin, Ireland
Antonia Azzini	Università degli Studi di Milano, Italy
Mauro Dragoni	Fondazione Bruno Kesler, Italy
Manfred Gilli	University of Geneva and Swiss Finance Institute, Switzerland
Philip Hamill	University of Ulster, UK

Serafin Martínez Jaramillo	Banco de México, Mexico
Youwei Li	Queen's University Belfast, UK
Christopher Clack	University College London, UK
José Ignacio Hidalgo	Universidad Complutense de Madrid, Spain
Malcolm Heywood	Dalhousie University, Canada
Enrico Schumann	VIP Value Investment Professionals
Piotr Lipinski	University of Wrocław, Poland
Ronald Hochreiter	WU Vienna University of Economics and Business, Austria
Ruppa Thulasiram	University of Manitoba, Canada
Wei Cui	University College Dublin, Ireland
Michael Mayo	University of Waikato, New Zealand
Jose Pinto	Instituto das Telecomunicacoes (IST/IT), Spain
Nikos Thomaidis	University of the Aegean, Greece
Eva Alfaro	Instituto Tecnológico de Informática, Spain
Krzysztof Michalak	Wrocław University of Economics, Wrocław, Poland

EvoGAMES Program Committee:

Lurdes Araújo	UNED, Spain
Simon Colton	Imperial College London, UK
Ernesto Costa	Universidade de Coimbra, Portugal
Francisco Fernández de Vega	Universidad de Extremadura, Spain
Leo Galway	University of Ulster, UK
Mario Giacobini	University of Torino, Italy
Johan Hagelbäck	Blekinge Tekniska Högskola, Sweden
John Hallam	University of Southern Denmark, Denmark
Pier Luca Lanzi	Politecnico di Milano, Italy
Tobias Mahlmann	IT-Universitetet i København, Denmark
Mike Preuss	University of Dortmund, Germany
Noor Shaker	IT-Universitetet i København, Denmark
Moshe Sipper	Ben-Gurion University, Israel
Julian Togelius	IT-Universitetet i København, Denmark
Georgios Yannakakis	University of Malta, Malta

EvoIASP Program Committee:

Antonia Azzini	University of Milan-Crema, Italy
Lucia Ballerini	University of Dundee, UK
Leonardo Bocchi	University of Florence, Italy
Oscar Cordón	University of Granada, 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	Ernst-Moritz-Arndt-Universität Greifswald, Germany
Francesco Fontanella	University of Cassino, Italy
Şpela Ivekoviç	University of Strathclyde, UK
Mario Koeppen	Kyushu Institute of Technology, Japan
Krisztof Krawiec	Poznan University of Technology, Poland
Jean Louchet	INRIA, France
Evelyne Lutton	INRIA, France
Pablo Mesejo Santiago	University of Parma, Italy
Luca Mussi	Henesis srl, Italy
Youssef Nashed	University of Parma, Italy
Ferrante Neri	University of Jyväskylä, Finland
Gustavo Olague	CICESE, Mexico
Riccardo Poli	University of Essex, UK
Sara Silva	INESC-ID Lisboa, Portugal
Stephen Smith	University of York, UK
Giovanni Squillero	Politecnico di Torino, Italy
Kiyoshi Tanaka	Shinshu University, Japan
Andy Tyrrell	University of York, UK
Roberto Ugolotti	University of Parma, Italy
Leonardo Vanneschi	Universidade Nova de Lisboa, Portugal

EvoINDUSTRY Program Committee:

María Arsuaga-Ríos	CERN, Switzerland
Anna I Esparcia-Alcázar	S2 Grupo, Spain
William B. Langdon	University College London, England, UK
John Levine	University of Strathclyde, Scotland, UK
Rhyd Lewis	University of Cardiff, Wales, UK
Ender Özcan	University of Nottingham, England, UK
A. Şima Etaner-Uyar	Istanbul Technical University, Turkey

EvoNUM Program Committee:

Anne Auger	INRIA, France
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Hans-Georg Beyer	FH Vorarlberg, Austria
Ying-ping Chen	National Chiao Tung University, Taiwan
Nikolaus Hansen	INRIA, France
José Ignacio Hidalgo	Universidad Complutense de Madrid, Spain
William B. Langdon	University College London, England, UK
Salma Mesmoudi	Institut des Systèmes Complexes - Paris Île-de-France, France

Boris Naujoks	University of Dortmund, Germany
Ferrante Neri	University of Jyväskylä, Finland
Gabriela Ochoa	University of Stirling, Scotland, UK
Petr Pošik	Czech Technical University, Czech Republic
Mike Preuss	University of Dortmund, Germany
Günter Rudolph	University of Dortmund, Germany
Ivo Fabian Sbalzarini	Max Planck Institute of Molecular Cell Biology and Genetics, Germany
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
A. Şima Etaner-Uyar	Istanbul Technical University, Turkey
Darrell Whitley	Colorado State University, USA

EvoPAR Program Committee:

Una-May O'Reilly	MIT, USA
Gianluigi Folino	ICAR-CNR, Italy
Jose Carlos Ribeiro	Instituto Politécnico de Leiria, Portugal
Garnett Wilson	Afinin Labs Inc., Canada
Malcolm Heywood	Dalhousie University, Canada
Kalyan Veermachneni	MIT, USA
Juan L. Jiménez	Université du Luxembourg
William B. Langdon	University College London, England, UK
Denis Robilliard	Université du Littoral-Côte d'Opale, France
Marco Tomassini	Université de Lausanne , Switzerland
José Ignacio Hidalgo	Universidad Complutense de Madrid, Spain
Leonardo Vanneschi	Universidade Nova de Lisboa, Portugal

EvoRISK Program Committee:

Hussein Abbass	UNSW, Australian Defence Force Academy, Australia
Robert K. Abercrombie	Oak Ridge National Laboratory, USA
Rami Abielmona	University of Ottawa, Canada
Anas Abou El Kalam	École Nationale Supérieure d'Ingénieurs de Bourges, France
Nabendu Chaki	University of Calcutta, India
Sudip Chakraborty	Valdosta State University, USA
Mario Cococcioni	NATO Undersea Research Centre, Italy
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Spain
Solange Ghernaouti-Hélie	University of Lausanne, Switzerland

Malcom Heywood	Dalhousie University, Canada
Miguel Juan	S2 Grupo, Spain
David Megías	Universitat Oberta de Catalunya, Spain
Javier Montero	Universidad Complutense de Madrid, Spain
Frank W. Moore	University of Alaska Anchorage, USA
Kay Chen Tan	National University of Singapore
Vicenç Torra	CSIC, Spain
Antonio Villalón	S2 Grupo, Spain
Xin Yao	University of Birmingham, UK
Nur Zincir-Heywood	Dalhousie University, Canada

EvoROBOT Program Committee:

Nicolas Bredèche	ISIR/UPMC-CNRS, France
Jeff Clune	University of Wyoming, USA
Stéphane Doncieux	ISIR/UPMC-CNRS, France
Marco Dorigo	Université Libre de Bruxelles, Belgium
Heiko Hamann	Karl-Franzens-Universität Graz, Austria
Giorgos Karafotias	Vrije Universiteit Amsterdam, The Netherlands
Jean-Marc Montanier	TAO, Université Paris-Sud XI, INRIA, France
Jean-Baptiste Mouret	ISIR/UPMC-CNRS, France
Stefano Nolfi	ICST-CNR, Italy
Claudio Rossi	Universidad Politécnica de Madrid, Spain
Sanem Sariel	Istanbul Technical University, Turkey
Florian Schlachter	University of Stuttgart, Germany
Thomas Schmickl	Karl-Franzens-Universität Graz, Austria
Christopher Schwarzer	Eberhard-Karls-Universität Tübingen, Germany
Jürgen Stradner	Karl-Franzens-Universität Graz, Austria
Jon Timmis	University of York, England, UK
Berend Weel	Vrije Universiteit Amsterdam, The Netherlands
Alan Winfield	University of the West of England, Bristol, UK

EvoSTOC Program Committee:

Jürgen Branke	University of Warwick, UK
Ernesto Costa	Universidade de Coimbra, Portugal
A. Şima Etaner-Uyar	Istanbul Technical University, Turkey
Yaochu Jin	University of Surrey, UK
Changhe Li	China University of Geosciences, China
Jorn Mehnen	Cranfield University, UK
Trung Thanh Nguyen	Liverpool John Moores University, UK
David Pelta	Universidad de Granada, Spain
Hendrik Richter	Leipzig University of Applied Sciences, Germany
Philipp Rohlfshagen	University of Essex, UK

Briseida Sarasola	Universidad de Málaga, Spain
Anabela Simões	Coimbra Polytechnic, Portugal
Renato Tinós	Universidade de São Paulo, Brazil
Krzysztof Trojanowski	Polish Academy of Sciences, Poland
Shengxiang Yang	De Montfort University, UK

General Track Program Committee:

Marco Gaudesi	Politecnico di Torino, Italy
Ernesto Sánchez	Politecnico di Torino, Italy
Alberto Tonda	Institut des Systèmes Complexes, France

Sponsoring Organizations

- Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology
- Institute for Informatics and Digital Innovation at Edinburgh Napier University, Scotland, UK
- The EvoCOMNET track was technically sponsored by the World Federation on Soft Computing

Table of Contents

EvoCOMNET

EvoCOMPLEX

EvoENERGY

EvoFIN

EvoGAMES

EvoIASP

EvoINDUSTRY

EvoNUM

EvoPAR

EvoRISK

EvoROBOT

EvoSTOC

An Evolutionary Framework for Routing Protocol Analysis in Wireless Sensor Networks

Doina Bucur[2,1], Giovanni Iacca[1], Giovanni Squillero[3], and Alberto Tonda[4,⋆]

[1] INCAS[3]
Dr. Nassaulaan 9, 9401 HJ, Assen, The Netherlands
giovanniiacca@incas3.eu
[2] Johann Bernoulli Institute, University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands
d.bucur@rug.nl
[3] Politecnico di Torino
Corso Duca degli Abruzzi 24, I-10129, Torino, Italy
giovanni.squillero@polito.it
[4] INRA UMR 782 GMPA
1 Avenue Lucien Brétignières, 78850, Thiverval-Grignon, France
alberto.tonda@grignon.inra.fr

Abstract. Wireless Sensor Networks (WSNs) are widely adopted for applications ranging from surveillance to environmental monitoring. While powerful and relatively inexpensive, they are subject to behavioural faults which make them unreliable. Due to the complex interactions between network nodes, it is difficult to uncover faults in a WSN by resorting to formal techniques for verification and analysis, or to testing. This paper proposes an evolutionary framework to detect anomalous behaviour related to energy consumption in WSN routing protocols. Given a collection protocol, the framework creates candidate topologies and evaluates them through simulation on the basis of metrics measuring the radio activity on nodes. Experimental results using the standard Collection Tree Protocol show that the proposed approach is able to unveil topologies plagued by excessive energy depletion over one or more nodes, and thus could be used as an offline debugging tool to understand and correct the issues before network deployment and during the development of new protocols.

Keywords: Wireless Sensor Networks, Anomaly Detection, Network Efficiency, Routing Protocols, Evolutionary Algorithms.

1 Introduction

The escalation of complexity in modern systems makes proving their correct behaviour an increasingly hard task. *Formal verification techniques* require a human and computational effort that severely limits their applicability. Thus, formal verification methods are used in real-world problems for a subset of the

⋆ All authors contributed equally and their names are presented in alphabetical order.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 1–11, 2013.
© Springer-Verlag Berlin Heidelberg 2013

system's software or hardware components [3], when the task can be artificially constrained or when simplified models are employed, thus significantly impairing the breadth and usability of the results. *Wireless Sensor Networks* (WSNs) are an emblematic example: while each node in a network is a constrained embedded system, the quality of networking depends upon physical topology and environmental conditions. Thus, the number of configurations to test easily skyrockets. Recent related work in the area of formal verification for WSNs [11,15,2,12] succeeds in checking only *qualitative* properties (e.g., assertions), and may only report a small number of faulty topologies.

This paper presents an evolutionary approach to routing protocol analysis for WSNs, which has two unprecedented advantages in that it obtains (i) *quantitative* metrics for the quality of a WSN, and (ii) a large number of faulty WSN configurations, which allow for better generalization of the results.

An evolutionary core generates network topologies, optimizing special heuristic metrics, with the final objective to uncover latent problems in the routing protocol. In the experiments, we focus on the Collection Tree Protocol (CTP) [5] and its TinyOS [9] implementation; we use the TinyOS simulator TOSSIM [10] for the evaluation of a topology. The metrics we consider relate to the number of networking (i.e., radio transmission and reception) *system calls* on WSN nodes, as these form a metric which best correlates to energy depletion (and thus lifetime), independently of the hardware platforms deployed in the WSN. We then uncover a large family of WSN configurations over which CTP causes extremely high traffic, which would drastically diminish the lifetime of a real-world WSN deployment. Based on this, we define a metric, named *degree of disconnection*, which can predict the energy consumption of the protocol over a WSN topology.

The idea of applying evolutionary computation to verification and testing has been explored in several different domains. Preliminary experiments in [16] show that stochastic meta-heuristics are effective in locating the most promising parts of the search space to test complex software modules. A flight system software is verified in [13], where a genetic algorithm outperforms classical random testing techniques. In [4], the operating system of a mobile phone prototype is tested with evolved keyboard inputs, uncovering several power-related bugs.

The remaining of this paper is organized as follows. Section 2 gives a brief overview of system faults in WSNs, including matters of energy consumption. Section 3 describes our approach for detecting anomalous network behaviour in CTP. The experimental results obtained by means of the evolutionary approach are discussed in section 4. Finally, section 5 concludes based on this experience, and outlines future works.

2 Anomalous WSN Routing and Lifetime

A WSN is a distributed, wirelessly networked, and self-organizing system most often employed for distributed monitoring tasks. Sensing nodes deployed at locations of interest sense, store and send data to one or more *sink* nodes for collection; in turn, a sink may disseminate commands back to the nodes. Network-layer

routing protocols for WSNs most often aim at organizing the nodes into a multi-hop tree-like logical topology for this collection and/or dissemination. Among collection protocols, the Collection Tree Protocol [5] is the *de facto* standard; we analyze its behaviour in this paper.

Given the uncontrolled, distributed nature of a WSN deployment, hardware, software, and networking faults will impact the network's performance; of these, networking anomalies are the most computationally intensive (and thus challenging) to verify against. Field reports [6,7,1] list a number of anomalies which unexpectedly impaired deployments, ranging from hardware clock drifts on individual nodes caused by a different response to ambient temperature, to nodes losing routes to the sink due to the network being unpredictably dense for the size of the routing table, and to unforeseen shortness of network lifetime.

An example of such a *lifetime anomaly* is reported in [7]: a link-layer protocol caused most sensor nodes in the deployment to run out of batteries within few weeks. Interestingly, the scientists were never certain *post factum* about the actual cause: *"Back-of-the-envelope calculations reveal that [..] another effect is causing nodes to consume more energy than anticipated. We observed that remote nodes lasted about one week longer. We conjecture that the difference is caused by overhearing less traffic (5 vs. 70 neighbors), but then the small (33%) difference in lifetime indicates that there must be another factor contributing significantly to the nodes' power consumption."* In this paper, we aim to uncover causes for such behavioural effects offline (at WSN design time), before deployment.

When analyzing a network protocol for the purpose of determining network configurations of worst lifetime, a good metric to represent energy depletion on the node is the number of system calls which execute radio receptions or transmissions. It is for radio operations that the battery current draw is highest on any given hardware platform; e.g., on the mainstream MicaZ sensor platform, 19.7 *mA* are drawn in radio-receive mode, compared to under 8 *mA* processor draw. In what follows, the radio system calls are our *events* of interest.

3 An Evolutionary Tool for WSN Routing Protocol Analysis

Uncovering a bug in a complex structure such as a WSN is like finding the proverbial "needle in a haystack": either a bug is discovered, or it is not, with no intermediate possibilities. Such a characteristic significantly impairs any evolutionary algorithm. In fact, the very idea of evolution is based on the accumulation of small differences, each one profitable from the point of view of the organism. In order to tackle such a difficult problem, the evolutionary algorithm is set to design a WSN configuration able to maximize the number of events raised during the simulation. Such a configuration is eventually used to highlight errors and issues. In the following, individual encoding and fitness functions are described.

3.1 Individual Description

For CTP verification, an individual represents a candidate configuration of the WSN, encoded as a square matrix $N \times N$, with N the number of nodes in the network. Each position i, j in the matrix holds the strength of the signal between nodes i and j, expressed in dBm. It is interesting to note that, given the nature of wireless transmitters and receivers, the matrix need not be symmetrical.

While a viable connection quality between two nodes can theoretically assume any value above -110 dBm (a threshold particular to TOSSIM), we reduce our problem to two subintervals of primary interest:

1. *strong links* with signal strength in the interval $SL = [-30, 0]$ dBm, over which small variations may well lead to different behaviours;
2. *weak links* with signal strength below -30 dBm, which we equate with no link at all.

We chose the interval SL as such in order to cap the effect that TOSSIM's noise model has upon network behaviour (see subsection 4.1), i.e., to confine the statistical variation among simulations of the same topology, and thus raise the confidence level of our evolutionary experiments (see subsection 4.2). Also, encoding instead the link quality as the integer interval [-31,0] dBm might force the evolutionary algorithm to a long exploration before discovering that results with a high number of weak links are interesting; on the other hand, employing an interval like [-60,0] dBm might lead to large groups of individuals with different genotype but exactly the same phenotype.

In order to help the algorithm explore the search space more effectively, individuals present two different *alleles* for each *gene* in their genome: a weak link, and a strong link with an integer strength in the interval SL. Since mutation operators can either change a gene to its allele or fine tune the strength, an individual with a weak link in a given matrix position is *close* to an individual with a strong link in the same position regardless of the actual strength of the link. A user-defined occurrence probability is associated to each allele.

3.2 Fitness Function

As reported above, evolutionary computation works best when there is a gradual slope towards the best solutions. Thus, if this is not present in the problem, it is crucial to artificially create such a slope in the fitness landscape. With this requirement in mind, two alternative metrics are adopted to identify different kinds of protocol anomalies. Following the intuitive idea that the more a device is stressed, the more likely it is to show possible problems, we maximize, in independent experiments, the following objectives:

1. `maxNetworkEvents`, the maximum number of node-local network system calls (including packet reception and packet transmission) among all nodes;
2. `sumNetworkEvents`, the sum of node-local network system calls (as above) on all the nodes in the network.

With the first fitness function, we aim at finding network configurations where at least one node generates an abnormal number of radio events, and thus will have diminished lifetime. With the latter, we aim at finding network topologies where a number of the nodes in the network cause a packet storm, which will lead to decreased lifetime for all involved. Based on our experimental results, these two fitness functions are proven smooth enough to distinguish between well-behaved and anomalous configurations.

3.3 μGP and TOSSIM

The algorithm used in the experiments is μGP, an evolutionary framework developed at Politecnico di Torino [14]. Two interesting properties influenced the choice of this particular evolutionary core. First, since μGP's individuals are represented as a succession of macros with a tunable probability of appearance, it is possible to describe an individual structure such as the one presented in subsection 3.1. Second, the design of the framework is based on the notion of an external evaluator, which makes the integration with a WSN simulator quite straightforward.

In order to test the candidate configurations, the open-source simulator TOSSIM [10] is chosen. TOSSIM is a discrete event simulator which allows users to test and analyze TinyOS sensor networks in a controlled and repeatable environment. Besides guaranteeing high fidelity simulations at different levels of granularity (from hardware interrupts to high-level system events), TOSSIM supports two programming interfaces (Python and C++) to control simulations and interact with them (even at runtime). In this work, the powerful scripting and parsing features of the Python interface are used to collect the number and type of different radio events raised by each node.

With reference to Fig. 1, the evolutionary core creates candidate configurations for TOSSIM. All the system calls on each node are logged during the simulation, and parsed successively to create an event hash map. This hash map is finally processed to obtain the fitness function, by simply counting the number of radio events raised by the nodes in the configuration.

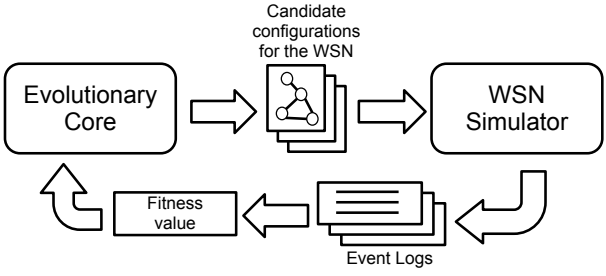


Fig. 1. Conceptual scheme of the proposed evolutionary approach

4 Experimental Results

In order to investigate different kinds of protocol anomalies, we ran experiments with three probabilities (25%, 50% and 75%) of having a strong link when generating a new connection (see subsection 3.1), and the two objectives described in subsection 3.2, namely `maxNetworkEvents` and `sumNetworkEvents`. Thus, six ⟨fitness function, strong link probability⟩ configurations were tested in total.

For simplicity, we consider topologies of 10 nodes booting at simulation time 0; there is enough randomness in the network stack for this setting not to cause network collisions due to synchronicity. Each topology is simulated for 200 seconds. Node 0 is the sink, while the others run CTP to form a multi-hop logical tree topology over the physical graph topology, for the purpose of data collection to the sink. In particular, each node samples one of its on-board sensors every second, and after bundling a number of readings, sends them to the sink.

4.1 Noise Analysis

WSNs are affected by a number of stochastic effects, including radio-frequency noise, interference among nodes and from external sources, packet collision, etc. TOSSIM provides an accurate model of the mainstream MicaZ radio stack. In addition to that, to further improve the realism of the simulation (e.g., by taking into account bursts of interference) it is possible to add a statistical *noise model* over the original links' signal strengths. This model is generated automatically from a trace measured experimentally in real-world testbeds, with the generation algorithm based on Closest Pattern Matching [8]. In our experiments, we used the `light-short` noise trace available with TOSSIM.

Several experiments were conducted in order to investigate the influence of the noise on the two fitness functions defined above. More specifically, for each fitness function and percentage of strong links, 200 topologies were sampled randomly in the topology design space, and each one was evaluated 50 times to compute the average fitness value and its variance σ^2. From these experiments it emerged that the variance of the noise dramatically changes depending on the specific topology, making the problem particularly challenging from an evolutionary point of view. Numerical results are presented in subsection 4.3, compared and contrasted with the results obtained with the evolutionary approach.

4.2 Evolutionary Process

A detailed exposition of the μGP evolutionary process is out of the scope of this paper. An interested reader may find all relevant information in [14]. μGP was executed with the parameter settings displayed in Table 1, two standard crossover operators (one-point and two-point), and mutation operators that act at the level of a single gene. The evolutionary algorithm was configured in such a way that a $\mu+\lambda$ strategy was used, together with a self-adaptation scheme on the activation of the different genetic operators. Each evolutionary process was allotted a computational budget of 24 hours. It should be noted that the wall

Table 1. μGP parameters setting

Parameter	Description	Value
`mu`	population size	40
`lambda`	number of genetic operators applied at every step	5
`inertia`	inertia for the self-adapting parameters	0.9
`sigma`	strength of genetic operators	0.9
`tau`	size of the tournament selection	2

Table 2. Configuration of the experiments considered in this study

Fitness function	Strong links	Generations	Evaluations	Best Fitness
`maxNetworkEvents`	25%	206	1231	13554.6
	50%	225	1392	13603.9
	75%	208	1371	12986.1
`sumNetworkEvents`	25%	168	1000	80574.8
	50%	176	1115	87431.1
	75%	221	1406	59995.9

clock time for each individual simulation heavily depends on the specific topology and the number of events it generates. Thus, the number of individuals evaluated during each experiment, which in turns affects the number of generations performed, is extremely variable, as shown in Table 2. Additionally, in order to compute a statistically meaningful average fitness value, also in this case each evaluated individual was simulated multiple times with different random seeds. Recalling that the standard error of the mean of a n-dimensional sample whose variance is σ is $\sigma/\sqrt{n}$, and applying the central limit theorem to approximate the sample mean with a normal distribution, it can be proved that a sample size $n = 16\sigma^2/W^2$ guarantees a 95% confidence interval of width W. Thus, in order to guarantee a confidence interval $W = \sigma$ regardless of the actual value of σ (which in any case is not constant in the search space, as we have seen before) we chose $n = 16$ simulations per topology.

4.3 Results: Faulty Topologies

We comparatively present top individuals obtained by the evolutionary algorithm, and random topologies evaluated separately from the evolutionary experiment; we only detail the experiments configured for 50% strong links, but similar considerations apply to the experiments with 25% and 75% strong links. We first analyze the *top individual* for `sumNetworkEvents`, aiming to derive an explanation for its radio behaviour; Fig. 2 (left) depicts the top individual as an *undirected graph*, computed from the original directed graph so that the undirected edge between nodes m, n exists iff both directed edges $m \to n$, $n \to m$ exist. This shows that the topology is *disconnected* with respect to undirected links; directional links may still exist between graph components, but they prove insufficient for a well-behaved topology.

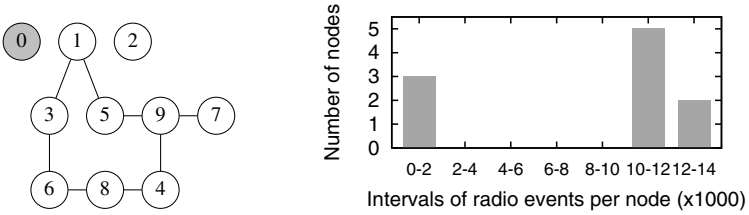


Fig. 2. (left) Undirected physical topology for `sumNetworkEvents` top individual with 50% strong links. (right) Histogram of radio events per node: the top two buckets contain nodes {1, 3, 4, 5, 6, 8, 9}, which form a cycle disconnected from the sink.

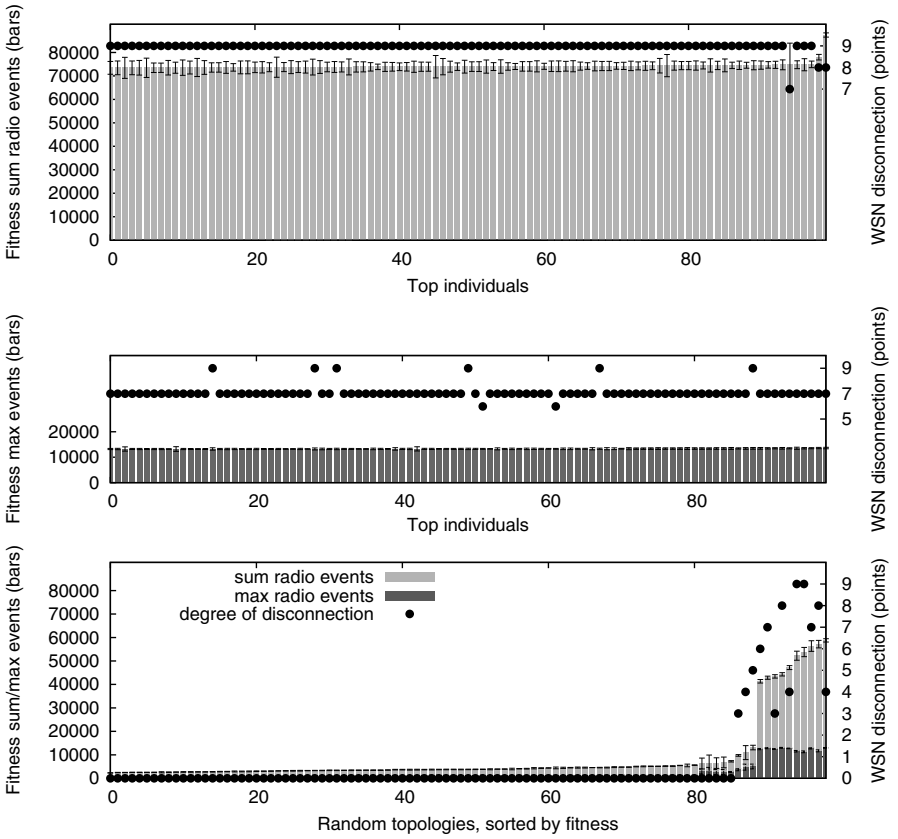


Fig. 3. Overview of fitness values (left y axis) for 100 top and random individuals, all with 50% probability of strong links. Superimposed, we show the degree of disconnection for each individual (right y axis).

Fig. 2 (right) gives the histogram of radio system calls per node in the network. This shows that not fewer than seven out of ten nodes nearly reach the maximum number of events per node; correlating this fact with the topology of the individual, we can state that a traffic storm is caused among nodes in a component disconnected from the sink.

On the basis of this observation, we define a *metric* for physical WSN topologies, which will qualitatively predict the values of both fitness functions. We call this *degree of disconnection*, and we define it over the undirected version of the topology graph G as the number of nodes in all those *graph components* of G which (i) do not include the sink node 0, and (ii) have size greater than 2 nodes:

$$\text{degree of disconnection}(G) := \sum_{\substack{C_i \in \text{ components}(G) \\ 0 \notin C_i,\ \text{size}(C_i) > 2}} \text{size}(C_i)$$

This metric is intuitive: it equals 0 for a *connected* graph G, and every graph component which both does not contain the sink node 0 and is large enough to form cycles contributes to the sum; small, 1- or 2-node components cause few radio events. For a given topology, we conclude that a degree of disconnection strictly greater than 0 is reason for concern with regard to the `maxNetworkEvents` fitness; the higher the value of this metric, the more `sumNetworkEvents` may rise. For the top individual in Fig. 2, this degree of disconnection is 8.

Fig. 3 gives the top 100 individuals and the same number of random topologies for both fitnesses, all with 50% strong links. The fitness value of a topology is shown as the average and standard deviation over 16 simulations (as introduced in subsection 4.2). All top individuals have a degree of disconnection of at least 6 (for the `maxNetworkEvents` fitness) and 7 (for `sumNetworkEvents`). As seen from our random testing, low fitness values correlate well with zero disconnection, which are both marks of well-behaved topologies.

When comparing random tests with our evolutionary algorithm, it becomes clear that the power of the evolutionary analysis is to uncover a large number of faulty topologies, all of higher fitness than found through random testing.

5 Conclusions and Future Works

Detecting anomalous behaviours in WSNs is a complex task, due to non-trivial interactions between nodes. This paper presented an innovative approach for analyzing the anomalous behaviour of routing protocols in WSNs. Following the intuitive concept that stressing a device is likely to uncover eventual anomalies, an evolutionary framework is set to generate network configurations, with the objective to maximize the number of events raised by the nodes. Experimental results demonstrate that the proposed approach is able to identify network layouts that generate undesired traffic storms, thus potentially being extremely useful to analyze and correct lifetime-related issues before network deployment and during the development of new protocols.

Future works will (i) explore the effectiveness of different fitness functions targeting other behavioural faults, e.g., the ratio of delivered data packets, and the number of packets received in duplicate at the sink; (ii) consider the use of a variable-length genome, including node reboots and other node-level events; and (iii) broaden the analysis to include other WSN protocols.

Acknowledgments. INCAS^3 is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional Development and the Ministry of Economic Affairs, Peaks in the Delta.

References

1. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M.: The hitchhiker's guide to successful wireless sensor network deployments. In: Proc. 6th ACM Conference on Embedded Network Sensor Systems, SenSys 2008, pp. 43–56. ACM (2008)
2. Bucur, D., Kwiatkowska, M.: On software verification for sensor nodes. Journal of Systems and Software 84(10), 1693–1707 (2011)
3. D'Silva, V., Kroening, D., Weissenbacher, G.: A survey of automated techniques for formal software verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(7), 1165–1178 (2008)
4. Gandini, S., Ruzzarin, W., Sanchez, E., Squillero, G., Tonda, A.: A framework for automated detection of power-related software errors in industrial verification processes. Journal of Electronic Testing 26(6), 689–697 (2010)
5. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys 2009, pp. 1–14. ACM, New York (2009)
6. Jurdak, R., Wang, X.R., Obst, O., Valencia, P.: Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies. In: Tolk, A., Jain, L.C. (eds.) Intelligence-Based Systems Engineering. ISRL, vol. 10, ch. 12, pp. 309–325. Springer, Heidelberg (2011)
7. Langendoen, K., Baggio, A., Visser, O.: Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In: Proc. Int. Conf. on Parallel and Distributed Processing, pp. 174–181. IEEE Computer Society (2006)
8. Lee, H., Cerpa, A., Levis, P.: Improving wireless simulation through noise modeling. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN 2007, pp. 21–30. ACM, New York (2007)
9. Levis, P., Gay, D., Handziski, V., Hauer, J.H., Greenstein, B., Turon, M., Hui, J., Klues, K., Sharp, C., Szewczyk, R., Polastre, J., Buonadonna, P., Nachman, L., Tolle, G., Culler, D., Wolisz, A.: T2: A second generation OS for embedded sensor networks. Tech. Rep. TKN-05-007, Technische Universität Berlin (2005)
10. Levis, P., Lee, N., Welsh, M., Culler, D.E.: TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In: Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 126–137 (2003)
11. Li, P., Regehr, J.: T-Check: bug finding for sensor networks. In: Proceedings of the 9th International Conference on Information Processing in Sensor Networks (IPSN), pp. 174–185. ACM (2010)
12. Mottola, L., Voigt, T., Österlind, F., Eriksson, J., Baresi, L., Ghezzi, C.: Anquiro: Enabling efficient static verification of sensor network software. In: Workshop on Software Engineering for Sensor Network Applications (SESENA) ICSE(2) (2010)

13. Sacco, G., Barltrop, K., Lee, C., Horvath, G., Terrile, R., Lee, S.: Application of genetic algorithm for flight system verification and validation. In: Aerospace Conference, pp. 1–7. IEEE (2009)
14. Sanchez, E., Schillaci, M., Squillero, G.: Evolutionary Optimization: the μGP toolkit, 1st edn. Springer Publishing Company, Incorporated (2011)
15. Sasnauskas, R., Landsiedel, O., Alizai, M.H., Weise, C., Kowalewski, S., Wehrle, K.: KleeNet: discovering insidious interaction bugs in wireless sensor networks before deployment. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 186–196. ACM (2010)
16. Shyang, W., Lakos, C., Michalewicz, Z., Schellenberg, S.: Experiments in applying evolutionary algorithms to software verification. In: IEEE World Congress on Computational Intelligence (CEC), pp. 3531–3536. IEEE (2008)

Routing Low-Speed Traffic Requests onto High-Speed Lightpaths by Using a Multiobjective Firefly Algorithm

Álvaro Rubio-Largo and Miguel A. Vega-Rodríguez

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

Abstract. Nowadays, the bandwidth requirements of the majority of traffic connection requests are in the range of Mbps. However, in optical networks each physical link is able to operate in the range of Gbps causing a huge waste of bandwidth as a result. Fortunately, using access station at each node of the optical network, several low-speed traffic requests may be multiplexed onto one high-speed channel. Multiplexing or grooming these low-speed requests is known in the literature as the Traffic Grooming problem - an NP-hard problem. Therefore, in this paper we propose the use of Evolutionary Computation for solving this telecommunication problem. The selected algorithm is an approach inspired by the flash pattern and characteristics of fireflies, the Firefly Algorithm (FA), but adapted to the multiobjective domain (MO-FA). After performing several experiments and comparing the results obtained by the MO-FA with those obtained by other approaches published in the literature, we can conclude that it is a good approach for solving this problem.

Keywords: Firefly Algorithm, Multiobjective optimization, Traffic Grooming, WDM optical networks.

1 Introduction

More and more people make use of the Internet; however, our current data networks do not present enough bandwidth for handling this exponential growth of traffic requests. Fortunately, the use of optical fiber has solved this drawback thanks to its enormous bandwidth.

Since the bandwidth of an optical fiber link is around 50Tbps and the average requirements of a traffic request is in the range of Mbps, a waste of bandwidth comes up. With the aim of wasting as little as possible, the Wavelength Division Multiplexing technology tries to divide each physical link into several wavelengths of light (λ) or channels [8]. In this way, the speed rate per channel is in the range of Gbps.

There still exists a waste of bandwidth at each end-to-end connection from one node to another over a specific wavelength of light (lightpath). To solve it,

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 12–21, 2013.
© Springer-Verlag Berlin Heidelberg 2013

the use of access stations at each node provides them the ability of grooming several low-speed requests onto high-speed lightpaths [8].

This process of grooming low-speed traffic requests is known in the literature as the Traffic Grooming problem, an NP-hard problem [9] which consists on three subproblems: *lightpath routing*, *wavelength assignment*, and *traffic routing*.

The Traffic Grooming problem has been tackled by other authors in the literature. Zhu and Mukherjee [9] presented a mathematical formulation of the Traffic Grooming problem, as well as two reference heuristics for solving this telecommunication problem: *Maximizing Single-hop Traffic* (MST) and *Maximizing Resource Utilization* (MRU). These heuristics have been considered by many other authors in order to prove the goodness of new proposals for the Traffic Grooming problem. A procedure with an integrated grooming algorithm based on an auxiliary graph model (INGPROC) is reported by Zhu et al. [7]. Finally, De et al. [1] described an algorithm to handle general multi-hop static Traffic Grooming based on the Clique Partitioning (TGCP) concept.

All the aforementioned approaches are single-objective methods that try to optimize the throughput of a given optical network. In recent literature, new methods have been proposed for solving this telecommunication problem by using Multiobjective Optimization. Prathombutr et al. [4] proposed a well-known Multiobjective Evolutionary Algorithm (MOEA), the Strength Pareto Evolutionary Algorithm (SPEA). Later, Rubio-Largo et al. [5] tackled the problem by applying innovative MOEAs. They propose the Differential Evolution with Pareto Tournaments (DEPT) algorithm, the Multiobjective Variable Neighborhood Search algorithm (MO-VNS), and two well-known algorithms: Fast Non-Dominated Sorting Genetic Algorithm (NSGA-II) and the Strength Pareto Evolutionary Algorithm (SPEA2).

In this work, we propose the use of a multiobjective version of a Swarm Intelligence Evolutionary Algorithm. The single-objective version is the Firefly Algorithm (FA) [6], a population-based approach inspired by the flash pattern and characteristics of fireflies. In order to prove the goodness of the Multiobjective FA (MO-FA), we present several comparisons with other approaches published in the literature.

The remainder of this paper is organized as follows. Section 2 is devoted to present the Traffic Grooming problem. In Section 3, we describe the Multiobjective Firefly Algorithm. A comparison with other approaches published in the literature is reported in Section 4. Finally, the conclusions and possible lines of future work are presented in Section 5.

2 Traffic Grooming Problem

In this work, an optical network topology is modeled as a directed graph $G=(N, E)$, where N is the set of nodes and E is the set of physical links connecting nodes.

In the first place, we list several assumptions taken into account in this work:

- The number of wavelengths per optical fiber link (W) will be the same for all links in E.
- For all links in E, the wavelength capacity (C) will be the same. Furthermore, for every physical link (m,n)$\in E$, where $m, n \in N$; the propagation delay (d_{mn}) is equal to 1.
- At each node of N, the number of incoming physical links will be identical to the number of outgoing physical links.
- The number of transceivers (T_i/R_i or simply T) will be greater than or equal to 1, for all nodes in N.
- None of the nodes in N supports the *wavelength conversion* facility. Thus, we assume the *wavelength continuity constraint* [3]: all lightpaths must use the same wavelength (λ) across all physical links that they traverse.
- We suppose a static traffic pattern, where the set of low-speed traffic requests are known in advance. The granularity of each low-speed request is OC-x (Optical Carrier transmission rate), $x \in$\{1, 3, 12, and 48\}. Note that the bandwidth will be $x \times 51.84Mb/s$.
- We assume the *multi-hop grooming facility* [9]: a low-speed connection can use several concatenated lightpaths for being established. In addition, the low-speed requests cannot be split into several lower speed connections and routed separately.

In the second place, we briefly describe the other related parameters of the traffic grooming problem:

- P_{mn}, $\forall m, n \in N$: number of fibers interconnecting nodes m and n.
- Λ: traffic demand matrix

$$\Lambda^x = [\ \Lambda^x_{sd};\ s, d \in N\ ]_{|N|\times|N|},$$

  where Λ^x_{sd} is the number of OC-x connection requests between node pair (s, d).
- V^w_{ij}, $\forall w \in \{1 \dots W\}, \forall i, j \in N$: number of lightpaths from node i to node j on wavelength w (Virtual Topology).
- $P^{ij,w}_{mn}$, $\forall w \in \{1 \dots W\}, \forall i, j, m, n \in N$: number of lightpaths between node i to node j routed through fiber link (m, n) on wavelength w (Physical Topology route).
- S^x_{sd}, $x \in \{1, 3, 12,$ and $48\}, \forall s, d \in N$: number of OC-x streams requested from node s to node d that are successfully routed.

Hence, given an optical network topology, a fixed number of transmitters and receivers at each node, a fixed number of available wavelengths per fiber, a capacity of each wavelength, and a set of connection requests with different bandwidth granularity, the Traffic Grooming problem may be stated as a Multiobjective Optimization Problem (MOOP) [2], in which our objective functions are:

– *Traffic Throughput* (f_1): Maximize the total successfully routed low-speed traffic demands on virtual topology.

$$\begin{array}{c} Max \ \ \sum_{s=1}^{|N|} \sum_{d=1}^{|N|} \sum_{t=1}^{\Lambda_{sd}^{x}} (x \times S_{sd}^{x,t}) \\ x \in \{1, 3, 12, \ and \ 48\} \end{array} \tag{1}$$

– *Number of Transceivers or Ligthpaths* (f_2): Minimize the total number of transceivers used or the total number of lightpaths established.

$$Min \ \ \sum_{i=1}^{|N|} \sum_{j=1}^{|N|} \sum_{w=1}^{W} V_{ij}^{w} \tag{2}$$

– *Average Propagation Delay* (APD, f_3). Minimize the average hop count of lightpaths established, because to we assume $d_{mn} = 1$ in all physical fiber links (m, n).

$$Min \ \ \frac{\sum_{i=1}^{|N|} \sum_{j=1}^{|N|} \sum_{m=1}^{|N|} \sum_{n=1}^{|N|} (d_{mn} \times \sum_{w=1}^{W} P_{mn}^{ij,w})}{\sum_{i=1}^{|N|} \sum_{j=1}^{|N|} \sum_{w=1}^{W} V_{ij}^{w}} \tag{3}$$

For a complete formulation of the Traffic Grooming problem, including all parameters, variables, constraints, objective functions, and an illustrative example, please see [5].

3 Multiobjective Firefly Algorithm (MO-FA)

A population-based evolutionary algorithm inspired by the flash pattern and characteristics of fireflies is proposed by Yang in [6], the Firefly Algorithm (FA). This innovative approach uses the bioluminescent aptitudes of fireflies to attract other fireflies which are flying around. However, in this work, we propose a multiobjective version of the FA with the aim of dealing with the Traffic Grooming problem. The representation of a firefly (individual) is determinant for understanding how the algorithm tackles the problem. In this way, the chromosome representation adopted in this work is the same as in [5]. The chromosome is represented as a $|N| \times |N|$ matrix, where $|N|$ is the number of nodes in the given network topology. At each position of this matrix, we find a vector which stores T transmitters, and a weighting factor (WF_{ij}). So, for each pair of nodes, we can establish a maximum of T lightpaths. A more detailed description about the chromosome is given in [5], as well as a procedure for generating random individuals. The MO-FA works as follows:

Input:
- $G(N, E)$: physical topology
- W: number of wavelengths per link
- T: number of transceivers per node
- C: capacity of each wavelength
- Λ: traffic demand matrix
- K: number of shortest paths
- NF: Number of Fireflies in the population
- β_0: attractiveness
- γ: absorption coefficient
- α: control parameter for exploration
- *Stopping Criterion*

Output:
- PF: set of non-dominated solutions

Step 1) Initialization:

Step 1.1) Set PF as empty. $PF = \emptyset$

Step 1.2) Compute the K shortest paths between any two nodes m,n in N if and only if $m \neq n$, and $\Lambda^x_{mn} > 0$ for all $x \in \{1, 3, 12, \text{and } 48\}$. The set of K shortest paths for all nodes will be stored in $K - sp$.

Step 1.3) Generate NF random fireflies/individuals $x_1, x_2, ..., x_{NF}$

Step 2) Update:

For i,j = 1,...,NF do

Step 2.1) If and only if x_j attracts/dominates x_i ($x_j \succ x_i$), then move x_i towards x_j by using the following equation:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha \left(rand[0, 1] - \frac{1}{2} \right) \tag{4}$$

Step 2.2) Update the objective functions of x_i (f_1, f_2, and f_3) by using $G(N, E)$, W, T, C, Λ, and $K - sp$.

Step 2.3) Update the set of non-dominated solutions (PF) with x_i. Add the firefly x_i to PF only if no other firefly dominates it. Then, remove from PF all fireflies dominated by x_i.

Step 3) Stopping Criterion: If the stopping criterion is satisfied, then stop and output PF. Otherwise, go to Step 2.

As we may observe, we start initializing the set of non-dominated solutions as empty. Then, we calculate the K shortest paths for each pair of nodes under certain constraints (see Step 1.2) and we generate NF random fireflies. In the original FA (single-objective version), those fireflies with better value of fitness are considered more attractive to the rest of fireflies flying around. In our multiobjective version of the FA, we have used the concept of *Dominance* ($\succ$) [2] to

decide whether a firefly attracts another firefly or not (see Step 2.1). In the case of a firefly x_j attracts or dominates x_i, the firefly x_i moves towards x_j according to equation 4. In equation 4, r_{ij} refers to the Euclidean distance between x_i and x_j. Furthermore, in this equation, the second term corresponds with the attraction, and the third one is devoted to add dispersion to the algorithm. In this work, we assume the same factor of attractiveness as in [6], β_0=1. Only in the case of moving x_i towards x_j, we update the objective functions of x_i, as well as updating the set of non-dominated solutions (Steps 2.2 and 2.3). Finally, if the stopping criterion is not satisfied, then we go to Step 2. Otherwise, we return the set of non-dominated solutions (PF).

The firefly algorithm is very similar to the Particle Swarm Optimization algorithm (PSO). In fact, according to [6], the limiting case $\gamma = 0$ corresponds to the standard PSO. In [6], Yang presented a comprehensive comparison between the FA and PSO in order to prove the differences between the two algorithms.

4 Experimental Results

In this section, we present a comparison between the MO-FA and other approaches published in the literature. In the first place, we introduce the methodology followed to perform the comparisons.

In the comparisons, on the one hand, we compare the MO-FA with the MOEAs proposed in [5] by using multiobjective metrics. On the other hand, we make a single-objective comparison between the MO-FA and other methods published in the literature.

To carry out the comparison with the DEPT, MO-VNS, NSGA-II, and SPEA2, we have used a 6-node network topology (see [5]). This network consists of 6 nodes and 16 physical links with a capacity (C) equals to OC-48. Furthermore, in order to demonstrate the effectiveness of the MO-ABC, we have 40 different scenarios in which we vary the available resources per node and per link: T={3,4,5,6,7,8,9,10,11,12} and W={1,2,3,4}. The set of low-speed traffic request is the same as in [5], with a total amount of traffic of 988 OC-1 units. All the MOEAs were run using g++ (GCC) 4.4.5 on a 2.3GHz Intel PC with 1GB RAM. Furthermore, the number of independent runs was 30 for each data set, where the stopping criterion is based on the runtime: 30s.

The configuration used for DEPT, MO-VNS, NSGA-II, and SPEA2 is presented in [5]. In the case of the MO-FA, we have adjust the parameters of the algorithm. To decide the best value of each parameter, we have performed 30 independent runs and measured the HV of each run. In this way, we compute the median HV in the 30 runs, and select the value with the highest median. Next, we present the values tested, where we have highlighted in bold the value with the highest median of HV for each parameter:

- Number of Fireflies (NF): 25, 50, 75, **100**, 125, 150, 175, 200.
- Attractiveness (β_0): 0.05, 0.1, 0.25, 0.5, 0.75, **1**.
- Absorption coefficient (γ): 0.05, 0.1, 0.25, 0.5, 0.75, **1**.
- Control parameter for exploration (α): 0.05, 0.1, **0.25**, 0.5, 0.75, 1.

Table 1. Average HV obtained by the approaches. The notation used for pointing the statistically non-significant differences between pairs of algorithms is the following: (*) DEPT and MO-VNS, (**) DEPT and MO-FA, (†) MO-VNS and SPEA2, (‡) MO-VNS and MO-FA, and (§) NSGA-II and SPEA2.

T	W	DEPT	MO-VNS	NSGA-II	SPEA2	MO-FA	
3	1	34.96%	35.21%	33.35%	32.33%	35.42%	
4	1	39.29%	40.01%	36.48%	36.58%	41.28%	§
5	1	41.80%	42.67%	38.87%	38.79%	45.00%	§
6	1	43.45%	44.39%	39.84%	40.24%	47.45%	
7	1	44.61%	45.61%	39.99%	41.26%	49.18%	
8	1	45.48%	46.53%	42.76%	42.01%	50.47%	
9	1	46.88%	46.57%	43.66%	42.37%	48.18%	
10	1	47.44%	47.11%	43.91%	42.83%	48.79%	
11	1	47.90%	47.55%	43.61%	43.20%	49.29%	
12	1	48.28%	47.91%	43.64%	43.52%	49.71%	§
3	2	38.02%	37.93%	36.05%	35.50%	37.40%	*
4	2	47.03%	46.53%	44.46%	43.66%	47.33%	
5	2	52.60%	51.95%	50.29%	49.29%	53.31%	
6	2	56.41%	55.35%	53.17%	52.51%	57.16%	
7	2	58.97%	57.74%	56.41%	54.79%	59.95%	
8	2	60.87%	59.52%	57.82%	56.48%	62.03%	
9	2	62.30%	60.89%	59.42%	57.79%	63.63%	
10	2	63.44%	61.99%	60.48%	58.84%	64.91%	
11	2	64.37%	62.88%	61.40%	59.69%	65.95%	
12	2	65.14%	63.63%	62.21%	60.40%	66.82%	
3	3	37.95%	38.16%	36.16%	35.73%	38.40%	
4	3	48.89%	48.50%	46.64%	45.88%	49.07%	**
5	3	56.95%	55.52%	54.41%	53.07%	57.17%	
6	3	61.40%	59.88%	59.33%	57.45%	62.33%	
7	3	64.81%	63.36%	62.40%	60.68%	66.38%	
8	3	67.35%	65.95%	65.52%	63.09%	69.41%	
9	3	69.22%	68.90%	67.36%	65.04%	72.19%	
10	3	70.80%	70.46%	68.67%	66.54%	74.08%	
11	3	72.09%	71.73%	70.47%	67.74%	75.62%	
12	3	73.16%	72.79%	71.07%	68.78%	76.91%	
3	4	38.27%	38.20%	36.20%	35.69%	37.92%	*
4	4	49.02%	48.58%	46.38%	45.95%	48.60%	‡
5	4	58.22%	56.56%	55.82%	54.95%	57.75%	
6	4	64.54%	60.93%	61.75%	60.87%	64.19%	†
7	4	69.04%	64.31%	66.33%	64.89%	68.75%	
8	4	72.36%	67.05%	68.88%	67.87%	72.15%	
9	4	74.93%	68.78%	71.40%	70.17%	74.78%	**
10	4	76.98%	70.99%	73.91%	72.00%	76.87%	**
11	4	78.64%	71.71%	73.63%	73.50%	78.58%	**, §
12	4	80.03%	72.80%	76.93%	74.74%	79.99%	**
$\overline{HV}$		57.35%	55.93%	54.53%	53.42%	58.61%	

In the first place, we compare the approaches by using the well-known *Hypervolume* (HV) quality indicator [11]. Since this metrics is not free from arbitrary scaling of objectives, we have to normalize the non-dominated solutions obtained by the MOEAs before calculating the HV. In this work, the normalization points depend on the data sets. Therefore, for each scenario, the maximum values for each objective are f_1=*Total Amount of Traffic*, f_2=$|N| * T$, and f_3=$|N|$; and the minimum values are f_1=1, f_2=1, and f_3=1. For example, in a data set with T=4, the non-dominated solutions will be normalized by using the points (988,24,6) and (1,1,1); respectively. Note that, the normalization points are different than in [5]. In Table 1, we present a comparison among DEPT, MO-VNS, NSGA-II, SPEA2, and MO-FA by using the average value of HV obtained by each MOEA in 30 independent runs. As we may observe, the MO-FA obtains higher values of HV in most of the data sets. In Figure 1, we present an illustrative comparison of the MOEAs. As we can see, the MO-FA is clearly better than the rest of approaches when the resources in the optical networks are limited (W=1, W=2, and W=3); however, it performs similar to the DEPT algorithm in the easiest scenarios (W=4). In order to ensure

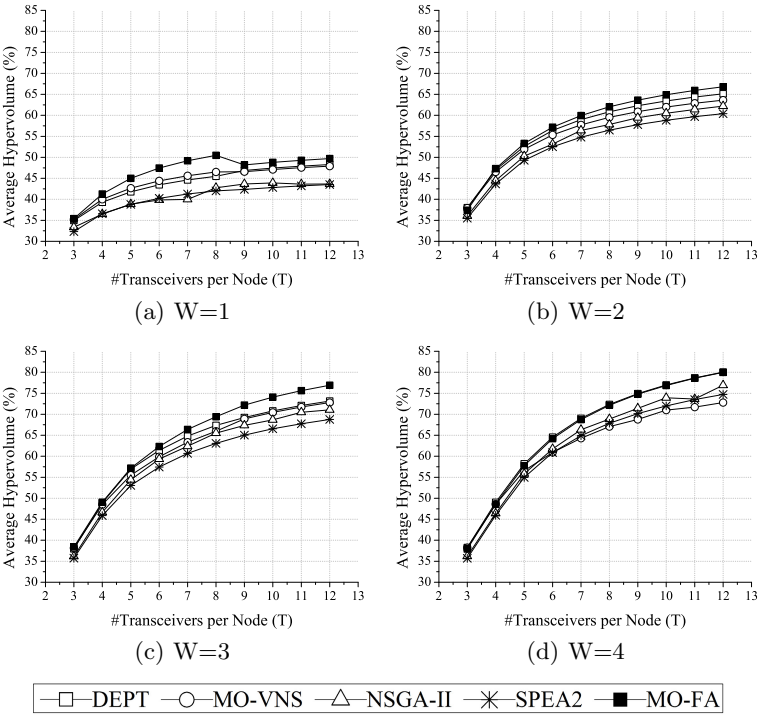


Fig. 1. Illustrative comparison among the MOEAs using the average Hypervolume in 30 runs

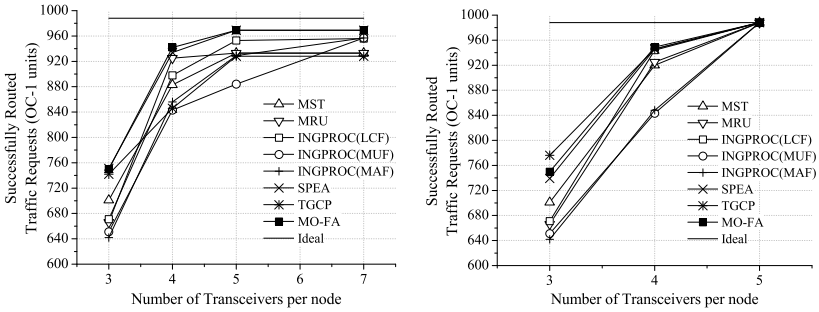
a certain level of confidence in the comparison, we have performed the same statistical analysis by pair of MOEAs than in [5], with a confidence level of 95%. In Table 1, we only report those data sets in which the differences of HV between two algorithms are statistically not significant.

Secondly, we compare the MOEAs by using the *Set Coverage* (SC) indicator [10], which measures the fraction of non-dominated solutions achieved by an algorithm B; which are covered by the non-dominated solutions achieved by an algorithm A (A$\succeq$B). In Table 2, we present the average percentage of dominance between each pair of MOEAs with different number of wavelengths per link. As we may observe, the MO-FA is able to dominate the vast majority of the non-dominated solutions obtained by any other MOEA. In contrast, the other algorithms only cover a low percentage of the solutions obtained by the MO-FA.

To perform the single-objective comparison with other heuristics published in the literature, we use only seven out of the forty scenarios (we have not found results of the other heuristics for the rest of scenarios): T=3 W=3, T=4 W=3, T=5 W=3, T=7 W=3, T=3 W=4, T=4 W=4, and T=5 W=4. In this comparison, we compare the maximum throughput (y_1) obtained by each approach in each scenario. We compare the MO-FA with the following methods: Maximizing

Table 2. Average SC by pair of MOEAs (A$\succeq$B)

A	B	W=1	W=2	W=3	W=4	$\overline{SC}$
DEPT	MO-VNS	79.20%	71.29%	64.23%	65.94%	70.16%
	NSGA-II	97.08%	98.26%	94.69%	96.45%	96.62%
	SPEA2	96.21%	97.24%	98.87%	92.79%	96.28%
	MO-FA	28.65%	25.38%	16.46%	14.86%	21.34%
MO-VNS	DEPT	31.90%	28.53%	31.40%	30.01%	30.46%
	NSGA-II	81.94%	70.37%	67.07%	63.34%	70.68%
	SPEA2	79.64%	72.36%	68.15%	62.53%	70.67%
	MO-FA	26.74%	21.71%	15.13%	16.33%	19.97%
NSGA-II	DEPT	7.26%	3.59%	4.77%	4.89%	5.13%
	MO-VNS	16.01%	10.46%	15.28%	10.81%	13.14%
	SPEA2	51.61%	53.41%	55.37%	44.31%	51.18%
	MO-FA	5.03%	2.67%	2.62%	2.43%	3.19%
SPEA2	DEPT	5.60%	1.77%	3.12%	2.33%	3.21%
	MO-VNS	12.83%	5.34%	12.26%	8.04%	9.62%
	NSGA-II	41.23%	25.77%	29.01%	35.73%	32.94%
	MO-FA	2.98%	1.40%	2.12%	1.90%	2.10%
MO-FA	DEPT	88.78%	81.02%	90.89%	92.04%	88.18%
	MO-VNS	93.17%	87.44%	90.48%	92.74%	90.96%
	NSGA-II	99.12%	100%	99.23%	99.86%	99.55%
	SPEA2	96.21%	100%	99.84%	100%	99.01%



(a) W=3 (b) W=4

Fig. 2. Illustrative comparison between the MO-FA and several approaches published in the literature

Single-Hop Traffic (MST) [9], Maximizing Resource Utilization (MRU) [9], INtegrated Grooming PROCedure (INGPROC) [7] with several traffic policies (*Least Cost First* (LCF), *Maximum Utilization First* (MUF), and *Maximum Amount First* (MAF)), Strength Pareto Evolutionary Algorithm (SPEA) [4], and the Traffic Grooming based on Clique Partitioning (TGCP) [1]. In Figure 2, we can see that the values of throughput obtained by the MO-FA are higher than or equal to those ones obtained by other methods in almost all data sets.

5 Conclusions and Future Work

In this work, we have presented a Multiobjective Evolutionary Algorithm inspired by the flash pattern and characteristics of fireflies for solving a telecommunication problem, the Traffic Grooming problem.

After performing several experiments, we can say that the MO-FA obtains higher quality results than the MOEAs published in [5]. Furthermore, we can

conclude that the MO-FA not only obtains more than one solution per run (multiobjective algorithm), optimizing three objective functions; but is also able to obtain results better than or equal to other single-objective approaches that only optimize one objective.

As future work, we intend to apply the MO-FA to the Traffic Grooming problem by using large network topologies with the aim of proving the effectiveness of our proposal.

Acknowledgements. This work was partially funded by the Spanish Ministry of Economy and Competitiveness and the ERDF (European Regional Development Fund), under the contract TIN2012-30685 (BIO project). Álvaro Rubio-Largo is supported by the research grant PRE09010 from Gobierno de Extremadura (Consejería de Economía, Comercio e Innovación) and the European Social Fund (ESF).

References

1. De, T., Pal, A., Sengupta, I.: Traffic Grooming, Routing, and Wavelength Assignment in an Optical WDM Mesh Networks Based on Clique Partitioning. Photonic Network Communications 20, 101–112 (2010)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York (2001)
3. Gagnaire, M., Koubaa, M., Puech, N.: Network Dimensioning under Scheduled and Random Lightpath Demands in All-Optical WDM Networks. IEEE Journal on Selected Areas in Communications 25(S-9), 58–67 (2007)
4. Prathombutr, P., Stach, J., Park, E.K.: An Algorithm for Traffic Grooming in WDM Optical Mesh Networks with Multiple Objectives. Telecommunication Systems 28, 369–386 (2005)
5. Rubio-Largo, A., Vega-Rodríguez, M.A., Gomez-Pulido, J.A., Sanchez-Perez, J.M.: Multiobjective Metaheuristics for Traffic Grooming in Optical Networks. IEEE Transactions on Evolutionary Computation, 1–17 (2012) (available online since June 2012)
6. Yang, X.-S.: Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009)
7. Zhu, H., Zang, H., Zhu, K., Mukherjee, B.: A Novel Generic Graph Model for Traffic Grooming in Heterogeneous WDM Mesh Networks. IEEE/ACM Transaction on Networking 11, 285–299 (2003)
8. Zhu, K., Mukherjee, B.: A Review of Traffic Grooming in WDM Optical Networks: Architectures and Challenges. Optical Networks Magazine 4(2), 55–64 (2003)
9. Zhu, K., Mukherjee, B.: Traffic Grooming in an Optical WDM Mesh Network. IEEE Journal on Selected Areas in Communications 20(1), 122–133 (2002)
10. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8, 173–195 (2000)
11. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)

Pareto-optimal Glowworm Swarms Optimization for Smart Grids Management

Eleonora Riva Sanseverino[1], Maria Luisa Di Silvestre[1], and Roberto Gallea[2]

[1] DIEETCAM, Universita' di Palermo, Viale delle Scienze, Palermo, Italy
{eleonora.rivasanseverino,marialuisa.disilvestre}@unipa.it
[2] DICGIM, Universita' di Palermo, Viale delle Scienze, Ed.6, Palermo, Italy
roberto.gallea@unipa.it

Abstract. This paper presents a novel nature-inspired multi-objective optimization algorithm. The method extends the glowworm swarm particles optimization algorithm with algorithmical enhancements which allow to identify optimal pareto front in the objectives space. In addition, the system allows to specify constraining functions which are needed in practical applications. The framework has been applied to the power dispatch problem of distribution systems including Distributed Energy Resources (DER). Results for the test cases are reported and discussed elucidating both numerical and complexity analysis.

Keywords: evolutionary optimization, swarm-optimization, pareto optimization, micro-grids.

1 Introduction

The management of modern electrical distribution systems has become complex due to the large penetration of Renewable Energy Sources (RES) and to the uncertain behavior of customers that can inject power in the network. These systems are referred to as 'smart grids', namely electricity networks that can intelligently integrate the actions of all users connected to it - generators, consumers and those that do both - in order to efficiently deliver sustainable, economic and secure electricity supplies ('European Technology Platform Smart Grid' definition). In these systems, an efficient energy management system and a proper regulation of some quantities, such as voltage and frequency, are required. Optimal energy management in these systems is typically performed on a 24-hours basis (a day-ahead) and can be regarded as optimally dispatching a set of Distributed Energy Resources of the grid (energy production units, energy storage systems and if possible, loads). Taking into account the former issue, the problem can be regarded as optimally dispatching a set of sources of the grid. Optimality is achieved minimizing quantities related to dispatch, which are namely *power loss (PL)*, *operational costs (OC)*, and sometime *carbon emissions (CE)*. Additional constraints need to be satisfied for a pattern to be eligible for dispatch. These may involve voltage drops limitations (VD) and currents in branches below their *ampacity*, i.e. ampere capacity (AC). For these

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 22–31, 2013.
© Springer-Verlag Berlin Heidelberg 2013

reasons, this optimization problem has a multi-objective nature and cannot be handled with classical analytical methods. Altough many efforts were done in dealing with this kind of problems, a golden standard does not exist and multi-objective optimization is still an open issue. Since the objectives set is composed of contrasting goals, a common way of solving multi-objective problem is to find the Pareto-optimal front of solutions, i.e. the set of equivalent non-dominated solutions w.r.t. to the given goals, provided that no preferences exist among the given goals. Some of the state-of-art methods are based on enhanced Genetic Algorithms [1,2]. In this paper we propose a novel extension of a recently proposed nature-inspired algorithm, the glowworm optimization, that can deal with pareto optimization. The method is applied to solve a *real* management problem, demonstrating how it is suitable for actual problems.

The paper is arranged as follows: in Section 2 some of the state-of-art multi-objective optimization methods based on pareto set approximation are illustrated. Then, in Section 3 the proposed algorithm is introduced providing both a theoretical explaination and technical details, while in Section 4 the details of the application to the distribution systems are elucidated. Section 5 shows the application of the method to the micro-grid optimal management to two actual test scenarios and the results are evaluated. Finally, in Section 6, final considerations are taken and future directions are pointed out to the reader.

2 Related Works

Multi-objective optimization is far from being (relatively) as straightforward as single-objective optimization. The definition itself of a "solution" for this kind of problems is not univocal. Usually, the solution to a multi-objective optimization is considered as approximating or computing all or a representative set of Pareto optimal solutions [3,4]. Pareto Optimal solutions set is defined as the set of solution vectors with the properties that none of the objective functions can be improved in value without impairment in some of the other objective values. In absence of any other preference criteria, all of the vectors in the Pareto Set can be considered equivalent from a purely mathematical perspective. Many endeavours have been done towards Pareto optimization using a wide range of approaches. All of these methods lie in different taxonomies, among these *scalarization*, which reduces the given problem to the optimization of a single goal, composed as a linear combination of the given objectives (Eq. 1).

$$\min_{x \in X} \sum_{i=0}^{k-1} w_i f_i(\mathbf{x}). \tag{1}$$

Such simple approach in some cases can warrant to achieve convergence to the actual Pareto set, but often has the drawback that it is very hard or impossibile to find the correct weighting parametrs w_i to reach convergence. In addition this approach suffers from objective scaling.

Another class of methods is the so-called *no-preference* methods, which, provided that all of the objectives have equal importance, reduce the problem to the minimization of the following function:

$$\min \left\| f\left(\mathbf{x}\right) - \mathbf{z}^{\mathbf{ideal}} \right\|, \tag{2}$$

where $\|\cdot\|$ can be any L_p norm. This method is also known as *global criterion* [5].

Methods which require additional knowledge given by experts are known as *apriori* methods, where objectives preference is expressed *before* the actual optimization process. Among these exist the *utility function method*, the *lexicographic method*, and *goal programming* [6].

The last category presented comprises the *a posteriori* methods, which produce a subset of the actual Pareto set. Among the most popular a posteriori methods, evolutionary algorithms have demonstrated to be suitable, such as *Non-dominated Sorting Genetic Algorithm-II (NSGA-II)* [1] and *Strength Pareto Evolutionary Algorithm 2 (SPEA-2)* [2].

The following paper hybridizes the non-domination sorting concept of NSGA-II with a traditionally single objective method, the *glowworm swarm optimization* algorithm (GSO). This synthesis is explained in the following paragraphes, after a detailed explaination of the original methods.

Classic Glowworm algorithm. In GSO [7], a swarm of agents are initially randomly distributed in the search space. Agents are modelled after glow-worms. Accordingly, they carry a luminescent quantity called luciferin along with them. The glow-worms emit a light whose intensity is proportional to the associated luciferin and interact with other agents within a variable neighbourhood. In particular, the neighbourhood is defined as a local-decision domain that has a variable neighbourhood range r_{id} bounded by a radial sensor range r_s ($0 < r_{id} \leq r_s$. A glow-worm i considers another glow-worm j as its neighbour if j is within the neighbourhood range of i and the luciferin level of j is higher than that of i.

The decision domain enables selective neighbour interactions and aids information of disjoint sub-swarms. Each glow-worm is attracted by the brighter glow of other glow-worms in the neighbourhood. Agents in GSO depend only on information available in their neighbourhood to make decisions. Each glow-worm selects, using a probabilistic mechanism, a neighbour that has a luciferin value higher than its own and moves toward it. These movements, that are based only on local information and selective neighbour interactions, enable the swarm of glow-worms to partition into disjoint subgroups that steer toward, and meet at, multiple optima of a given multimodal function. The GSO algorithm starts by placing a population of n glow-worms randomly in the search space so that they are well dispersed. Initially, all the glow-worms contain an equal quantity of luciferin l_0. Each iteration consists of a luciferin-update phase followed by a movement phase (update-position) based on a transition rule.

Luciferin-Update Phase: The luciferin update depends on the function value at the glow-worm position. During the luciferin-update phase, each glow-worm adds, to its previous luciferin level, a luciferin quantity proportional (using the

luciferin enhancement constant λ) to the fitness of its current location in the objective function domain. Also, a fraction of the luciferin value (the quantity $(1 - \rho$ is multiplied by the old value of luciferin and $\rho \leq 1)$ is subtracted to simulate the decay in luciferin with time.

Update-Position Phase: During the update position phase, each glow-worm decides, using a probabilistic mechanism, to move toward a neighbour that has a luciferin value higher than its own. The amplitude of the movement is proportional to the step size s. That is, glow-worms are attracted to neighbours that glow brighter. Further details can be found in [7].

NSGA-II. Non-dominated sorting algorithm II [1] is based on the concept of non-dominance of the solutions. The core of the methods, which is basically a classic evolutionary (genetic) algorithm is the fitness function definition. At each iteration, the current solutions are ranked according to a non-domimance criterion. Solutions which are globally non-dominated are assigned to rank 1, solutions which are dominated by rank 1 solutions only are assigned to rank 2, and so on, until the whole solution set is covered. The sorting rank is then used as a fitness value to the following genetic operators (selection, cross-over, etc.). In addition, during ranking a second parameter is computed for each individual, the *crowding distance*, which measures its proximity to its neighbors. Large average crowding distance results in a wider spread of the values in the population, thus diversification of the solutions.

3 Methods: GSO Extension to Pareto Optimization

GSO algorithm is intrinsically single-objective. To allow MO, a strategy to embed all of the objective values in a single quality parameter is required. Our choice for achieving MOGSO is to plug a non-dominated sorting criterion to filter the objectives vectors and rank them into Pareto fronts [1]. The achieved behavior is that, at each iteration, lower-rank swarm particles move towards top-rank particles with a velocity in direct proportion to their rank itself (i.e. the better the solution, the lesser its velocity). Thus at each iteration, $v_{rank1} < V_{rankj}$, where v_{rank1} is the velocity of solutions belonging to the first front of non dominated solutions and v_{rankj} is the velocity of solutions belonging to the j-th front of non dominated solutions (which is worst than rank 1). As a result, all of the glowworms will eventually lie in the top rank front, approximating the optimal Pareto front as required. Such approach has been investigated and tested for mathematical test problems and simple toy examples in our previous work [8].
Constraints handling. In addition to the basic formulation of the MOGSO approach, in order to use it in an actual problem, a constrainints handling mechanism is required. Solutions which violates constraints are not candidates for the final Pareto front, even if they are optimal from a objective optimization perspective. For this reason a simple further step is added to the main algorithm. This function checks the constraints, and swarm particles that result positive in their violation, are automatically assigned the lowest rank, notwithstanding

their objective values. This warrants that inadmissible solutions are discarded as they are spotted, moving the swarm particles away from their current position. Finally, the complete proposed MOGSO algorithm is summarized in the pseudocode of Algorithm 1. In the pseudo-code λ is the luciferin enhancement constant, a real number which scales the objective function J, and β is a parameter that affects the rate of change of the neighborhood range. J is the objective function which is as larger as lower the rank of the solution is. If constraints are violated J expresses the highest rank. The step $\ell_i(t)$ refers to the luciferin update, while the step $x_i(t+1)$ refers to the position updare of each glowworm. $N_i(t)$ represents the neighborhood of the i-th glowworm. n_t is a parameter indicating the neighborhood threshold.

Data:
m = number of dimensions;
n = number of glowworms;
s = step size;
x_i(t) = location of glowworm i at time t;
deploy-agents-randomly;
for $i = 1$ *to* n **do**
  $\ell_i(0) = \ell_0$
end
$r_d^i(0) = r_0$;
$t = 1$;
while $(t \leq itermax)$ **do**
  J = *non-dominated-sort*;
  J = *constraints-check*;
  for *each glowworm* i **do**
    $\ell_i(t) = (1-\rho)\,\ell(t-1) + \lambda J\left(x_i(t)\right)$;
  end
  for *each glowworm* i **do**
    $N_i(t) = \left\{j : d_{ij}(t) < r_d^i(t)\,;\, \ell_i(t) < \ell_j(t)\right\}$;
    for *each glowworm* $j \in N_j(t)$ **do**
      $p_{ij}(t) = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \in N_i(t)} \ell_k(t) - \ell_i(t)}$;
    end
    j = select-glowworm(); $x_i(t+1) = \min\left\{r_s, \max\left\{0, r_d^i(t) + \beta\left(n_t - \left|N_i(t)\right|\right)\right\}\right\}$;
  end
  $t \leftarrow t+1$;
end

Algorithm 1. Pseudocode algorithm for MOGSO optimization

4 A Test Case: Distribution System Management

As introduced in Section 1, the issue of optimal power dispatch among *Distributed Energy Sources* (DER) is a multi-objective task, due to multiple goals to be attained (such as minimum power losses, operational cost and carbon emissions). From the analytical point of view, optimal power dispatch through Unit commitment in smart grid appears to be very complicated since it is highly nonlinear and several constraints are required to be met. Besides, the presence of real and reactive storage units strongly influences the possibility to dispatch power and to perform voltage and frequency regulation by controlling their status. The literature on the subject can be divided according to the various solution methods for the problem: techniques such as dynamic programming, simulated annealing, tabu search, fuzzy computing and genetic algorithms can be applied to solve the issue. The literature on the optimal dispatch of energy sources can also

be tackled from the point of view of the formulation of the optimization problem as well as of the analysis of particular technical constraints, such as generator ramp limits, or environmental constraints, such as carbon dioxide emissions targets. The work in [9] presents a comprehensive survey of the most interesting papers on the issue.

The regulatory frame considered is that of a private smart grid, where all generation units are owned by the same subject [10]; the smart grid can either be a portion of a distribution system.

For a given design configuration (size, type and location of DERs; size and location of capacitor banks), knowing the hourly upper and lower production limits of each DER and the hourly loading level of each bus of the electrical distribution network, the objectives to be achieved are:

- the minimization of the yearly overall production costs;
- the minimization of the yearly CO2 emissions,

where the independent optimization variables are the hourly power productions of the DERs.

Consider a n-bus smart grid system with: - N_{fix} load or generation nodes with fixed forecasted real and reactive power demands or injections;
- N_{DER} controllable DERs. The problem is that to identify the real valued vector identifying the operating points of the DERs in the network hour by hour of the N_{day} representative days of the different periods of the year. The vector has $N_{DER} \cdot 24 \cdot N_{day}$ real elements; a subset of it taken for a generic hour h of the generic day d takes the following form (see Eq. 3):

$$x_{h,d} = \left[P_1^{g,h,d,}, P_2^{g,h,d,}, \ldots, P_{N_{DER}}^{g,h,d,}\right], \tag{3}$$

the relevant reactive powers, $Q_j^{g,h,d}$, can either be deduced since the bus are voltage controlled (PV nodes) or can also be optimization variables, in case the generation buses are considered PQ nodes.

The whole vector x can thus be written as (Eq. 4):

$$x = \left[x_{1,1}, x_{2,1}, \ldots, x_{24,1}, x_{1,2}, x_{2,2}, \ldots, x_{24,2}, \ldots, x_{1,N_{day}}, x_{2,N_{day}}, \ldots, x_{24,N_{day}}\right], \tag{4}$$

subject to the following constraints:

1. the values of the controlled variables, namely the DERs power otuputs (both active and reactive), taking into account the required power reserve should lie in the range defined by an upper limit $P_{j\,\max}^g$ and a lower limit $P_{j\,\min}^g$, (see Eq. 5 and 6),

$$P_{j\,\min}^g \leq P_j^{g,h,d} \leq P_{j\,\min}^g, \tag{5}$$

$$Q_{j\,\min}^g \leq Q_j^{g,h,d} \leq Q_{j\,\min}^g, \tag{6}$$

   where $P_j^{g,h,d}$ and $Q_j^{g,h,d}$ represent respectively the active and reactive power production at hour h of day d of the j-th DER;
2. the solution must give raise at all nodes of voltage below a maximum limit $\Delta V_{\min} = 5\%$;

3. the solution must satisfy the constraint about power transfer limits in the network lines, this contraint is usually always satisfied in well designed networks, therefore, in this formulation, it will not be considered.

The issue is that of finding the feasible vector x optimizing the following criteria:
- Yearly joule losses in the system (Eq. 7):

$$O_1(x) = \sum_{d=1,N_{day}} \sum_{h=1,24} \sum_{i=1,nr} R_i \left(I_i^{h,d}\right)^2 \Delta t. \tag{7}$$

In this expression, the energy losses in the system are evaluated as the summation of the power Joule losses on each of the n_r branches multiplied by Δt which in this case is 1 hour. The quantities in round brackets are considered constant in the considered time interval. This quantity is calculated after the solution of the power flow set of equations, which depends on the power injections at the nodes.
- Yearly fuel consumptions cost (Eq. 8)

$$O_2(x) = \sum_{d=1,N_{day}} \sum_{h=1,24} \sum_{i=1,N_{DG}} C_{Pi} P_i^{g,h,d} \Delta t, \tag{8}$$

where $C_{Pi}(P)$ denotes the unitary fuel consumption cost of the i-th source, $P_i^{g,h,d}$ is the power output of the i-th source at hour h and day d, considered constant in time interval Δt (in this case equal to 1 hour).

The dependency of unitary cost of production takes into account the reduced efficiency of microturbines at low output power. Please check [11] to see what variability has been considered. Therefore, the formulated problem is that to determine the operating points of the DERs giving rise to a technical-economical optimum as a compromise between minimum cost operation and high quality service. Minimum cost operation is ensured if the overall fuel consumption is minimum.

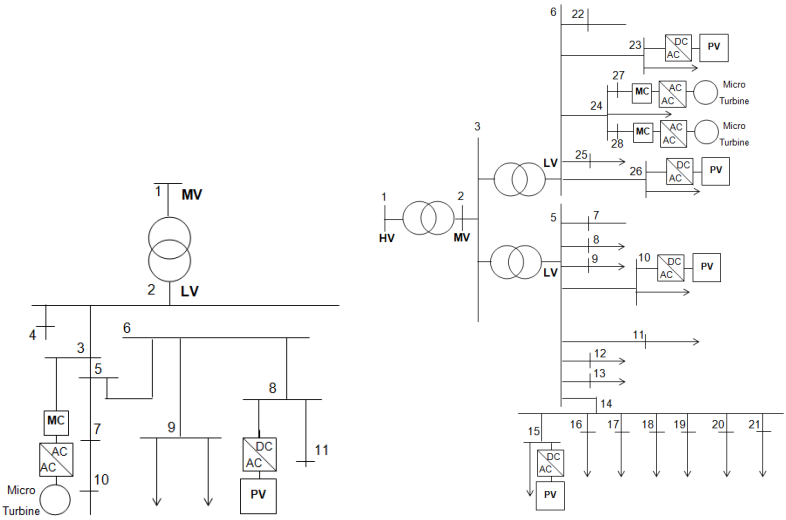
5 Results and Discussion

In order to test the MOGSO algorithm to optimal power dispatch, the method was applied to two actual test cases. For a more detailed theoretic validation, the reader is invited to consider the dissertation in [8].

Experimental Setup. Results section is dedicated to the minimization of production costs and power losses in smart grid. The applications are devoted to the minimization of the quantities in Equations 7 and 8 for a single day d and during 24 hours of operations. The tests have been carried out on two existing distribution test systems:

1. 11-buses smart grid, represented in Figure 1.a In the system there is one photovoltaic generation system and one microturbine;
2. 28-buses smart grid, represented in Figure 1.b. In the system there are four photovoltaic generation systems and two microturbines.

Simulations and Results Analysis. The simulations have been carried out on an AMD Phenom Quad-core at 2.30Ghz on Matlab platform, using the following parameters for the MOGSO algorithm: population = 50; iterations = 50;



(a) 11-buses smart grid (b) 28-buses smart grid

Fig. 1. Grid networks used for MOGSO testing applied to optimal power dispatch of DERs: 11-buses smart grid (a) and 28-buses smart grid (b)

the other parameters have been taken from the literature [12]. In average, the execution time $t_{e,GSO}$ of MOGSO is $\approx 10\%$ lower the calculation time $t_{e,NSGA-II}$ of NSGA-II. Figure 2 shows the approximated Pareto fronts for the considered test cases. As can be seen the solution found using MOGSO are slightly closer to the axis (i.e. better convergence to the actual Pareto front).

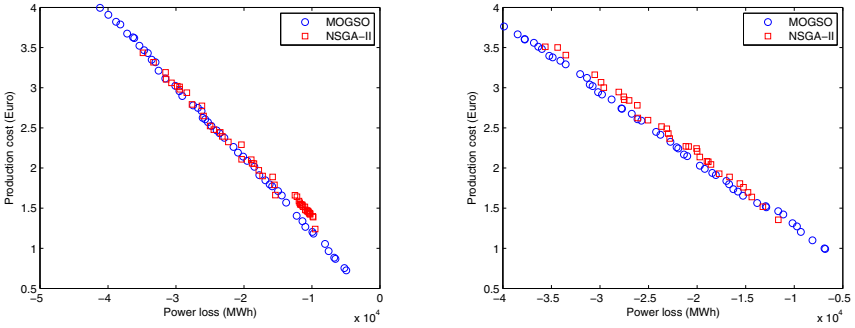
In addition to the convergence time other performance metrics were assessed. Due to the unavailability of the actual Pareto front for this problem, the spacing indicator $SP(S)$ has been used, which measures the uniform distribution of solution but which does not say anything about the number of solutions over the Pareto front, or its coverage. The formulation of the spacing index is reported in the following equations (Eq. 9 and 10).

$$SP(S) = \sqrt{1/|S-1| \sum_{i=1}^{|S|} \left(d_i - \overline{d}\right)}, \tag{9}$$

$$d_i = \min_{s_k \in S \cap s_k \neq s_i} \sum_{m=1}^{M} |f_m(s_i) - f_m(s_k)|. \tag{10}$$

where $\overline{d}$ is the average distance between points, calculated by the sum of absolute distances along each axis. We conducted 100 simulations for statistically assessing the SP of both *MOGSO* and *NSGA-II*, mean and variance values are reported in Table 1. Results show that the methods are comparable.

Complexity Analysis. In order to provide a complete dissertation about the proposed algorithm, a discussion about complexity is given. Each of the main part of the algorithm is analyzed from a computational complexity perspective:



(a) 11-buses smart grid (b) 28-buses smart grid

Fig. 2. Results for the two test cases: comparisons of the proposed MOGSO method with NSGA-II are shown for 11-buses smart grid (a) and 28-buses smart grid (b).

Table 1. Means and variances for the 100 simulations executed for the two test cases, both for *MOGSO* and *NSGA-II* algorithms

11-buses smart grid (Fig.1.a)		
Method	μ SP	σ^2 SP
MOGSO	$0.104 * 10^{-12}$	$7.525e * 10^{-27}$
NSGA-II	$0.089 * 10^{-12}$	$5.859e * 10^{-28}$

28-buses smart grid (Fig.1.b)		
Method	μ SP	σ^2 SP
MOGSO	$0.112 * 10^{-12}$	$7.349 * 10^{-28}$
NSGA-II	$0.095 * 10^{-12}$	$9.431 * 10^{-27}$

- *Luciferin-update phase*: this step is independent for each of the particles, so it is linear w.r.t the population size n (complexity $O(n)$).
- *Update-position phase*: this step is dependent on to every particle state, so, it is quadratic w.r.t the population size n (complexity $O(n^2)$).
- *Non-dominated sorting phase*: this step requires to make comparisons between al the possible combination of pairs of swarm particle n, thus requiring exactly n^2 comparisons (complexity $O(n^2)$).

As should be clear, the MO extension of GSO algorithm, does not introduce an increment in the complexity of the process, keeping it $O(n^2)$ regardless of the use of non domination sorting. In addition, an important point to mention is that, being a particle system, many parts of the method can take advantage of parallel hardware architectures such as Multi Core processing or gpGPU processing. Thus, a parallel implementation would drastically lower the computation time.

6 Conclusions and Future Work

In this paper a novel approach to multi-objective (Pareto) optimization was presented. It extends the traditional single-objective glowworm swarm optimization (GSO) algorithm by introducing some enhancements that allow to deal with multiple objectives and constraints. At each simulation step, the objective vector are

ranked according to their relative dominance. Glowworms particles are than directed to follow neighbors exhibiting highest dominance. As a result, at the end of the process, all of the particles lie on a representative subset of the complete Pareto front. The method has been applicated to the power distribution system in presence of DERs problem. In particular, two test cases were assessed, and the results are shown. In addition, an insight into algorithm complexity is provided. Further direction of research are the parallel implementation of the system and the introduction of mechanisms to give an higher diversity to the found solutions.

References

1. 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)
2. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou, K.C., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100 (2002)
3. Ehrgott, M.: Multicriteria Optimization. Lecture Notes in Economics and Mathematical Systems. Springer (2005)
4. Coello, C.A.C., Lamont, G.B., Van Veldhuisen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation Series. Springer (2007)
5. Hwang, C.L., Masud, A.S.M.: Multiple objective decision making, methods and applications: a state-of-the-art survey. Lecture Notes in Economics and Mathematical Systems. Springer (1979)
6. Charnes, A., Cooper, W.W.: Management models and industrial applications of linear programming. In: Management Models and Industrial Applications of Linear Programming, vol. 2, Wiley (1961)
7. Krishnanand, K.N., Ghose, D.: Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. Multiagent and Grid Systems 2(3), 209–222 (2006)
8. Riva Sanseverino, E., Di Silvestre, M.L., Gallea, R.: Multi-modal search for multiobjective optimization: an application to optimal smart grids management. In: MedPower. IET (2012)
9. Padhy, N.P.: Unit commitment-a bibliographical survey. IEEE Transactions on Power Systems 19(2), 1196–1205 (2004)
10. Vahedi, H., Noroozian, R., Hosseini, S.H.: Optimal management of microgrid using differential evolution approach. In: 2010 7th International Conference on the European Energy Market (EEM), pp. 1–6 (June 2010)
11. Graditi, G., Ippolito, M.G., Riva Sanseverino, E., Zizzo, G.: Optimal set points regulation of distributed generation units in micro-grids under islanded operation. In: 2010 IEEE International Symposium on Industrial Electronics (ISIE), pp. 2253–2260 (July 2010)
12. Zhang, H., Fu, P., Liu, Y.: Parameter settings analysis for glowworm swarm optimization algorithm. Journal of Information and Computational Science 9(11), 3231–3240 (2012)

An Overlay Approach for Optimising Small-World Properties in VANETs

Julien Schleich[1], Grégoire Danoy[1], Bernabé Dorronsoro[2], and Pascal Bouvry[1]

[1] Computer Science and Communications Research Unit
University of Luxembourg
{julien.schleich,gregoire.danoy,pascal.bouvry}@uni.lu
[2] Laboratoire d'Informatique Fondamentale de Lille
University of Lille 1
bernabe.dorronsoro_diaz@inria.fr

Abstract. Advantages of bringing small-world properties in mobile ad hoc networks (MANETs) in terms of quality of service has been studied and outlined in the past years. In this work, we focus on the specific class of vehicular ad hoc networks (VANETs) and propose to un-partition such networks and improve their small-world properties. To this end, a subset of nodes, called injection points, is chosen to provide back-end connectivity and compose a fully-connected overlay network. The optimisation problem we consider is to find the minimal set of injection points to constitute the overlay that will optimise the small-world properties of the resulting network, i.e., (1) maximising the clustering coefficient (CC) so that it approaches the CC of a corresponding regular graph and (2) minimising the difference between the average path length (APL) of the considered graph and the APL of corresponding random graphs. In order to face this new multi-objective optimisation problem, the NSGAII algorithm was used on realistic instances in the city-centre of Luxembourg. The accurate tradeoff solutions found by NSGAII (assuming global knowledge of the network) will permit to better know and understand the problem. This will later ease the design of decentralised solutions to be used in real environments, as well as their future validation.

Keywords: Multi-objective optimisation, VANETs, small-world.

1 Introduction

Mobile Ad hoc Networks (MANETs) are composed of mobile devices which spontaneously communicate with each other without any previously existing infrastructure. In MANETs, the limited radio range of the nodes, as well as their mobility, cause a highly fluctuating topology which can induce a severe degradation of the quality of service (QoS) or even lead to network partitioning, i.e., there is no available path between some pair of nodes. In order to leverage such issues many authors proposed to study or create small-world properties, i.e., high clustering coefficient and small mean-shortest path length, in wireless

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 32–41, 2013.
© Springer-Verlag Berlin Heidelberg 2013

networks as they are assumed to improve some QoS metrics, e.g., end-to-end throughput [13] or robustness to failure [5]. The main problem is however to find a practical way to establish any communication link in an ad hoc network, which is characterised by bounded transmission ranges. In this aim, we introduce the notion of injection points, which are nodes equipped with an additional communication interface. All injection points are assumed to be fully connected, i.e., any injection point can directly communicate with another one.

In this work, we focus on a special case of MANET called VANET, i.e., Vehicular Ad hoc NETworks, in which vehicles can either communicate with each other, in a peer to peer fashion, or with road-side units that allow access to backend systems. The scenario we will simulate for our experiments is the centre of Luxembourg city. The motivation for this scenario is mainly that Luxembourg city centre is covered by Wi-Fi access points, spread all over the city [1]. This pre-existing infrastructure will allow us to actually implement the notion of injection point in a later stage. The tackled problem is a three-objectives one, where (1) the number of injection points is to be minimised to limit communication overhead, and small-world properties are to be maximised, i.e., (2) the clustering coefficient (CC) is maximised so that it approaches the CC of a corresponding regular graph and (3) the difference between the average path length (APL) of the generated graph and the APL of corresponding random graphs is minimised.

The contribution of this paper is twofold. First, we propose a new paradigm based on an overlay-graph approach to cope with Watts original re-wiring process [19]. This new approach extends previous work [8,9,10] with a more realistic injection point model. Second, we tackle the problem in a multi-objective fashion with a well-known algorithm, namely NSGAII [11], in order to obtain a set of good compromise solutions. Therefore, we solve the problem in a centralised way, assuming that global knowledge of the network is known. The suitability of this approach for real world might be debeatable. It would require efficient mechanisms to communicate the network status to the server, as well as a fast optimisation algorithm to dynamically take decisions on the vehicles to use as injection points. However, it will definitely allow us to better understand the problem and find highly accurate tradeoff solutions for the three considered objectives. This is the aim of this work, because it will provide theoretical bounds that will be later used for the difficult task of designing decentralised protocols and validating them in reality, which will be our next step for future research.

The remainder of this paper is organised as follows. In the next section, a literature review on the usage of small-world properties in wireless networks in general and in MANETs in particular is provided. Then, Sect. 3 introduces the injection points problem and the corresponding optimisation model. The multi-objective algorithm used to tackle this problem is presented in Sect. 4. The experimental environment is described in Sect. 5 followed by the analysis of the obtained results in Sect. 6. Finally, the last section contains our conclusions and perspectives.

2 Related Work

Originally, the rewiring process proposed by Watts *et al.* to construct small-world networks [19] is intended to remove edges from a regular graph and replace them with new random edges. They demonstrated that it drastically reduces the average path length (APL), approaching the APL of random graphs, while keeping the clustering coefficient (CC) almost as high as in regular graphs.

In [14], the authors proposed the concept of contact nodes that are used by regular nodes to communicate with them when they need some network services. In this work, Helmy stated that contact nodes act as shortcuts but their implementation details are left for further research. Simulation results showed that only a fraction of nodes should endorse this role in order to obtain small-world properties.

Different contributions are investigating methods to choose which nodes are more suitable to act as contact nodes or which link should be deleted / rewired. Brust *et al.* proposed various decentralised algorithms to enhance small-world properties in MANETs. These include clustering algorithms to elect contact nodes [3], topology control algorithms that removes inefficient edges [4] or create new ones [5] to optimise the CC. In [2], the authors designed a framework for wireless networks to self-organise in a small-world fashion using only locally available information. It uses a non-random rewiring process to create useful shortcuts in wireless networks.

Other contributions focused on how to actually implement contact nodes and the rewiring process. Cavalcanti *et al.* first proposed in [6] to study the size of network nodes to be equipped with an additional long distance transceiver to effectively improve small-world properties. In [2], the authors proposed a similar idea but using directional beam forming. Another approach consists in increasing the transmission power of some nodes like in [5] and [18]. The results show that these mechanisms allow a reduction of the APL while keeping good values of CC.

Most recently, small-world properties have been studied for VANETs, in particular to design more realistic simulation environments. In [15], the authors studied the topology characteristics of VANETs and their implication on communication protocols design. In [17], the authors conducted a similar study on different networks and concluded that they already show some small-world properties. They developed a decentralised algorithm named *PSAMANET+* that makes use of the small APL and high CC information to efficiently broadcast information.

In this work, we have used realistic VANETs instances in the city centre of Luxembourg (see Sect. 5 for more details), in which the network is usually partitioned and the small-world properties can be greatly improved. Thus, we propose to use an existing infrastructure (Wi-Fi hotspots of Luxembourg City) in order to un-partition the network and enhance its characteristics.

3 Problem Description

The problem studied in this article consists in finding the best set of *injection points* to create a fully-connected overlay network that un-partitions a VANET

and maximises the resulting network small-world properties. Section 3.1 provides details on the small-world definition and used metrics, and Sect. 3.2 presents the tackled multi-objective optimisation problem.

3.1 Small-World Properties

In this optimisation problem, we consider small-world properties as indicators for the good set of rules to optimise the choice of injection points. Small-world networks [19] are a class of graphs that combines the advantages of both regular and random networks with respectively a high clustering coefficient (CC) and a low average path length (APL). The APL is defined as the average of the shortest path length between any two nodes in a graph $G = (V, E)$, that is $APL = \frac{1}{n(n-1)} \sum_{i,j} d(v_i, v_j)$ with $d(v_i, v_j)$ the shortest distance between nodes $v_i,\ v_j \in V$. It thus indicates the degree of separation between the nodes in the graph. The local CC of node v with k_v neighbours is $CC_v = \frac{|E(\Gamma_v)|}{k_v(k_v-1)}$ where $|E(\Gamma_v)|$ is the number of links in the relational neighbourhood of v and $k_v(k_v-1)$ is the number of possible links in the relational neighbourhood of v . The global clustering coefficient is the average of all local CC in the network, denoted as $CC = \frac{1}{n} \sum_v CC_v$. The CC measures to which extent strongly interconnected groups of nodes exist in the network, i.e., groups with many edges connecting nodes belonging to the group, but very few edges leading out of the group.

In this work, we consider Watts original definition of the small-world phenomenon in networks with $APL \approx APL_{random}$ and $CC \gg CC_{random}$, where APL_{random} and CC_{random} are, respectively, the APL and CC of random graphs with similar number of nodes and average node degree k.

3.2 Model of the Problem

This problem considers hybrid VANETs where each vehicle can potentially have both vehicle-to-vehicle and vehicle-to-infrastructure (e.g., using Wi-Fi hotspots) communications. Nodes elected as injection points (i.e., nodes connected to the infrastructure) form a fully connected overlay network, that aims at increasing the connectivity and robustness of the VANET. Injection points respectively permit to efficiently disseminate information from distant and potentially disconnected nodes and prevent costly bandwidth overuse with redundant information. An example network is presented in Fig. 1.

We consider small-world as the desirable network properties to reach and, as defined in the previous section, we rely on Watts definition of a small-world network by considering two metrics, i.e., $APL \approx APL_{random}$ and $CC \gg CC_{random}$. In addition, the number of chosen injection points has to be minimised as they may induce additional communication costs.

The proposed multi-objective optimisation problem can be formalised as follows. The solution to this problem is a binary vector s of size n (number of nodes in the network), $s[1..n]$ where $s[i] = 1$ if node v_i is an injection point, and $s[i] = 0$ if v_i is not an injection point. The decision space is thus of size 2^n.

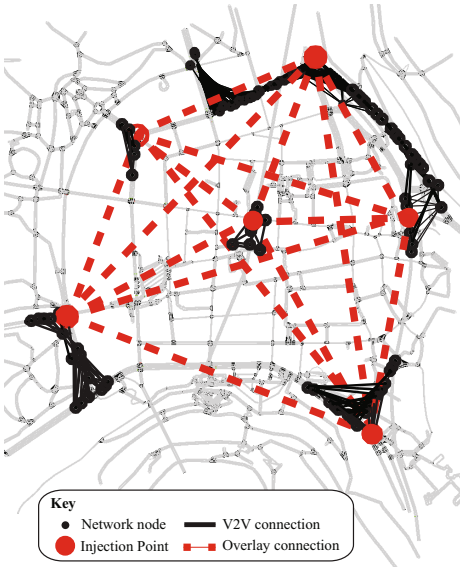


Fig. 1. Network with 248 nodes including 6 injections points composing the overlay network

This problem is a three objectives one, defined as:

$$f(s) = \begin{cases} min\,\{inj\} \\ max\,\{cc\}\ ; \\ min\,\{apl_{diff}\} \end{cases} \quad \text{s. t. } component = 1 \tag{1}$$

where inj is the number of chosen injection points, cc is the average clustering coefficient of the resulting network, and apl_{diff} is the absolute difference between the APL of the resulting network and the APL of the equivalent random graph (averaged over 30 different instances): $apl_{diff} = |apl - apl_{random}|$. These random graphs are generated using Watts rewiring process [19], i.e., with randomness $p = 1$. Since the initial objective is to unpartition the network, a constraint is set on the number of connected components in the created network, i.e. *component* must be equal to 1.

In order to optimise this hard three-objectives problem, we rely on the well-known multi-objective evolutionary algorithm, NSGAII, described in detail in the next section.

4 NSGAII

The NSGAII [11] algorithm is, undoubtedly, the reference algorithm in multi-objective optimisation. A pseudocode is given in Algorithm 1. NSGAII does not implement an external archive of non-dominated solutions, but the population itself keeps the best non-dominated solutions found so far. The algorithm starts

by generating an initial random population and evaluating it (lines 2 and 3). Then, it enters in the main loop to evolve the population. It starts by generating a second population of the same size as the main one. It is built by iteratively selecting two parents (line 6) by binary tournament based on dominance and crowding distance (in the case the two selected solutions are non-dominated), recombining them (we use the two-point crossover in our case) to generate two new solutions (line 7), which are mutated in line 8 (we use bit flip mutation) and added to the offspring population (line 9). The number of times this cycle (lines 5 to 10) is repeated is the population size divided by two, thus generating the new population with the same size as the main one. This new population is then evaluated (line 11), and merged with the main population (line 12). Now, the algorithm must discard half of the solutions from the merged population to generate the population for the next generation. This is done by selecting the best solutions according to ranking and crowding, in that order. Concretely, ranking consists on ordering solutions according to the dominance level into different fronts (line 13). The first front is composed by the non-dominated solutions in the merged population. Then, these solutions in the first front are removed from the merged population, and the non-dominated ones of the remaining solutions compose the second front. The algorithm iterates like this until all solutions are classified. To build the new population for the next generation, the algorithm adds those solutions in the first fronts until the population is full or adding a front would suppose exceeding the population size (line 14). In the latter case (lines 15 to 17), the best solutions from the latter front according to crowding distance (i.e., those solutions that are more isolated in the front) are selected to complete the population. The process is repeated until the termination condition is met (lines 4 to 18).

```
1:  //Algorithm parameters in 'nsga'
2:  InitialisePopulation(nsga.pop);
3:  EvaluatePopulation(nsga.pop);
4:  while ! StopCondition() do
5:     for index ← 1 to cga.popSize/2 do
6:        parents←SelectParents(nsga.pop);
7:        children←Crossover(nsga.Pc,parents);
8:        children←Mutate(nsga.Pm,children);
9:        offspringPop←Add(children);
10:    end for
11:    EvaluatePopulation(offspringPop);
12:    union←Merge(nsga.pop, offspringPop);
13:    fronts←SortFronts(union);
14:    (Pop', lastFront)←GetBestCompleteFronts(fronts);
15:    if size(nextPop) < nsga.popsize then
16:       Pop'←BestAccToCrowding(lastFront,nsga.popsize-size(Pop'));
17:    end if
18: end while
```

Algorithm 1. Pseudocode for NSGAII

5 Experimental Setup

We describe in this section the methodology we followed for our experiments. Solutions are represented as a binary string, every bit representing one vehicle.

Those genes set to 1 mean that the corresponding cars act as injection points, while a 0 value indicates the contrary.

The configuration used for NSGAII algorithm is the one originally suggested by the authors [11]. However, we had to adapt some parameters to deal with the binary representation of our problem (see Table 1): the two-point recombination and bit-flip mutation operators were used. In two-point recombination, two crossover positions are selected uniformly at random in both parents, and the values between these points are exchanged to generate two new offspring solutions. The bit-flip mutation is to change a 1 into a 0, or vice-versa. The algorithm evolves until $50,000$ fitness function evaluations are performed, and 30 independent runs were executed for every problem instance.

In terms of network, we have used realistic VANETs instances in the city centre of Luxembourg, simulated using the VehILux mobility model [16]. VehILux accurately reproduces the vehicular mobility in Luxembourg by exploiting both realistic road network topology (OpenStreetMaps) and real traffic counting data from the Luxembourg Ministry of Transport. The 4 studied networks represent snapshots of a simulated area of 0.6 km^2, 2 snapshots are taken between 6:00 a.m. and 6:15 a.m., instances and the 2 others between 7:00 a.m. and 7:15 a.m. These 4 network instances are named using their corresponding timestamp, starting from 21900 to 25800. In the case of Luxembourg city, this time range is characterised by a monotonously increasing number of vehicles due to a very dense commuting activity. The properties of the instances are shown in Table 2.

Table 1. NSGA-II configuration

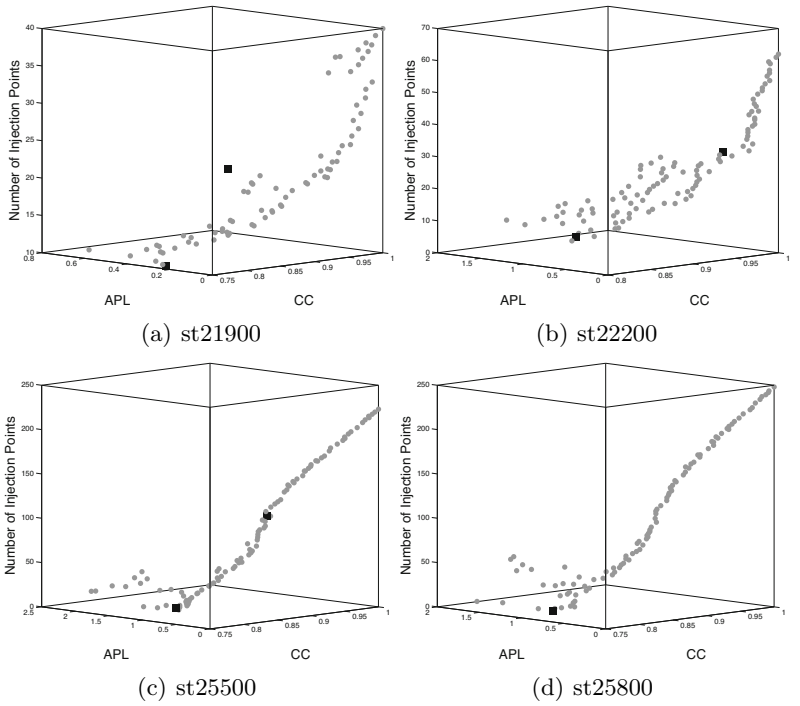
Parameters	Values
Population size	100
Final archive size	100
Max. evaluations	$50,000$
Pop. initialisation	Random
Selection	Binary tournament
Recombination	Two-point (DPX)
Probability	$p_c = 0.9$
Mutation	Bit-flip
Probability	$p_m = \frac{1}{ChromosomeLength}$
Independent runs	30

Table 2. Network instances

	Surface	0.6 km^2	
	Coverage radius	100 m	
6 a.m.	Network Number	21900	22200
	Number of Nodes	40	62
	Partitions	10	8
	Solution space	1^{12}	4.61^{18}
7 a.m.	Network Number	25500	25800
	Number of Nodes	223	248
	Partitions	10	6
	Solution space	1.34^{67}	4.52^{74}

6 Results

We present in this section the results we obtained in our experiments. They are plotted in Fig. 2. These Pareto fronts are obtained after merging all the non-dominated solutions reported by NSGAII in the 30 independent runs performed for every problem instance. In order to keep a subset of representative solutions (a maximum of 100 solutions are selected), the strength raw fitness technique of SPEA2 was used as density estimator to discard non-dominated solutions from the densest area. It was selected because it is more suitable than ranking and crowding for three dimensional problems [7].



(a) st21900 (b) st22200

(c) st25500 (d) st25800

Fig. 2. Pareto fronts found for the different problem instances

In Fig. 2, the black squares represent the best solutions we obtained in our previous work [12], using two panmictic GAs (generational and steady-state), a cellular GA, and a cooperative coevolutionary GA to solve an aggregative function of the objectives. Two different linear combinations were used for this single-objective approach. The first one only included the two small-world measures as a way to evaluate the number of injection points needed to reach optimal small-world values (referred to as no-inj from now), while the second one included all three objectives (that we call it inj). Each figure thus includes black squares that correspond to the best solution obtained with each objective function. As it can be seen, the solutions reported are well spread over a wide range of values. This is important to provide the decision maker with a large choice of diverse solutions to the problem. In these Pareto fronts, we observe that the higher the network density is, the lower the diversity of solutions found, especially for those solutions with high number of injection points. The reason is that dense networks have a small APL, allowing small improvements after adding injection points. Therefore, new injection points will mainly affect the CC value of the network.

Additionally, we analyse how good the multi-objective technique is performing compared to the solutions provided by the single-objective methods. First, we notice that the no-inj results are generally less accurate than their inj counterparts. Indeed, for the small instances, named 21900 and 22200, the no-inj solution

is dominated by 51.35%, resp. 4.41%, of the solutions in the Pareto front, while the inj solutions are non-dominated in most cases (only dominated 0%, resp. 0.44% of the solutions in the front). For these small instances, we can conclude that the multi-objective optimisation brings a substantial improvement as none of its solutions are dominated by the single objective techniques and some of them even dominate no-inj solutions. We would like to recall that the compared inj and no-inj solutions are the best results found after 30 independent runs of four different GAs [12]. In the case of the bigger instances, i.e., 25500 and 25800, the tendency for the inj single objective solution to be more accurate than its no-inj counterpart is confirmed. Indeed, the inj solution dominates 13.78%, resp. 26.69% of all the non-dominated solutions found in the 30 independent runs performed. While these results cannot be considered as flawless, it still means that 86.22%, resp. 73.31% of the solutions found are non-dominated and thus we propose a wide variety of good solutions to the decision maker. This is particularly true as the number of non-dominated solutions found increases with the size of the instance: 74, 227, 537, 532 for the considered instances, i.e., 21900, 22200, 25500, 25800. As a consequence, even if 74 solutions (26.69%) are dominated by the inj single objective solution in the 25800 instance, we still found 458 non-dominated solutions.

7 Conclusions and Future Work

We tackled in this work the un-partitioning of VANETs and their connectivity optimisation. For that, we proposed the use of road-side units to build an overlay network of backend connected nodes, called injection points, such that the small-world properties of the resulting VANET are optimised. We handle this new multi-objective problem with the well-known NSGAII algorithm to find highly accurate tradeoff solutions. The use of such algorithm using global knowledge of the network in reality might be arguable, because it would require the use of centralised servers to find solutions in real time, as well as high communication overheads. However, it will allow us to better know the problem and obtain theoretical bounds, with the aim of designing and validating novel decentralised protocols more suitable for real VANETs. A formulation of this novel multi-objective problem has been proposed, and two different scenarios representing realistic networks in the city centre of Luxembourg were used. Our experimental results show a plethora of different tradeoff solutions to the problem, offering a larger choice for the less dense networks studied. The solutions outperform, or are similar to, the most accurate ones we found in our previous work with four different genetic algorithms, using two biased single-objective formulations of the problem. Future work will exploit the obtained quasi-optimal results to evaluate the performances of decentralised approaches to elect injection points in order to reach unpartitioned networks with small-world properties in realistic VANETs.

Acknowledgments. B. Dorronsoro acknowledges the support by the Fonds National de la Recherche, Luxembourg (AFR contract no 4017742).

References

1. Hotcity network website, http://www.hotcity.lu
2. Banerjee, A., Agarwal, R., Gauthier, V., Yeo, C.K., Afifi, H., Lee, B.-S.: A self-organization framework for wireless ad hoc networks as small worlds. CoRR, abs/1203.1185 (2012)
3. Brust, M.R., Frey, H., Rothkugel, S.: Dynamic multi-hop clustering for mobile hybrid wireless networks. In: Int. Conf. on Ubiquitous Information Management and Communication (ICUIMC), pp. 130–135. ACM (2008)
4. Brust, M.R., Ribeiro, C.H.C., Turgut, D., Rothkugel, S.: LSWTC: A local small-world topology control algorithm for backbone-assisted mobile ad hoc networks. In: IEEE Con. on Local Computer Networks (LCN), pp. 144–151 (2010)
5. Brust, M.R., Turgut, D., Riberio, C.H.C., Kaiser, M.: Is the clustering coefficient a measure for fault tolerance in wireless sensor networks? In: IEEE Int. Conf. on Communications—Ad-hoc and Sensor Networking Symposium (ICC) (2012)
6. Cavalcanti, D., Agrawal, D., Kelner, J., Sadok, D.: Exploiting the small-world effect to increase connectivity in wireless ad hoc networks. In: IEEE Int. Conf. on Telecommunications (2004)
7. Coello Coello, C.A., Lamont, G.B., Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer (2007)
8. Danoy, G., Alba, E., Bouvry, P.: Optimal interconnection of ad hoc injection networks. Journal of Interconnection Networks (JOIN) 9(3), 277–297 (2008)
9. Danoy, G., Alba, E., Bouvry, P., Brust, M.R.: Optimal design of ad hoc injection networks by using genetic algorithms. In: Conf. on Genetic and Evolutionary Computation, GECCO, pp. 2256–2256. ACM (2007)
10. Danoy, G., Bouvry, P., Hogie, L.: Coevolutionary genetic algorithms for ad hoc injection networks design optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 4273–4280 (2007)
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evol. Comp. 6(2), 182–197 (2002)
12. Dorronsoro, B., Ruiz, P., Danoy, G., Pigné, Y., Bouvry, P.: Evolutionary Algorithms for Mobile Networks. Wiley (in press, 2013)
13. Filiposka, S., Trajanov, D., Grnarov, A.: Analysis of small world phenomena and group mobility in ad hoc networks. In: Innovative Algs. and Techns. in Automation, Industrial Electronics and Telecommunications, pp. 425–430. Springer (2007)
14. Helmy, A.: Small worlds in wireless networks. IEEE Communications Letters 7(10), 490–492 (2003)
15. Pallis, G., Katsaros, D., Dikaiakos, M.D., Loulloudes, N., Tassiulas, L.: On the structure and evolution of vehicular networks. In: Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecom. Systems, pp. 1–10. IEEE (2009)
16. Pigné, Y., Danoy, G., Bouvry, P.: A Vehicular Mobility Model Based on Real Traffic Counting Data. In: Strang, T., Festag, A., Vinel, A., Mehmood, R., Rico Garcia, C., Röckl, M. (eds.) Nets4Trains/Nets4Cars 2011. LNCS, vol. 6596, pp. 131–142. Springer, Heidelberg (2011)
17. Rezende, C., Boukerche, A., Pazzi, R.W., Rocha, B.P.S., Loureiro, A.A.F.: The impact of mobility on mobile ad hoc networks through the perspective of complex networks. J. Parallel Distrib. Comput. 71(9), 1189–1200 (2011)
18. Stai, E., Karyotis, V., Papavassiliou, S.: Socially-inspired topology improvements in wireless multi-hop networks. In: IEEE Int. Conf. on Communications (ICC), pp. 1–6 (2010)
19. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature 393(6684), 440–442 (1998)

Impact of the Number of Beacons in PSO-Based Auto-localization in UWB Networks

Stefania Monica and Gianluigi Ferrari

Wireless Ad-hoc and Sensor Networks Laboratory
Department of Information Engineering
University of Parma, I-43124 Parma, Italy
stefania.monica@studenti.unipr.it
gianluigi.ferrari@unipr.it
http://wasnlab.tlc.unipr.it/

Abstract. In this paper, we focus on auto-localization of nodes in a static wireless network, under the assumption of known position of a few initial nodes, denoted as "beacons". Assuming that Ultra Wide Band (UWB) signals are used for inter-node communications, we analyze the impact of the number of beacons on the location accuracy. Three different approaches to localization are considered, namely: the Two-Stage Maximum-Likelihood (TSML) method ; the Plane Intersection (PI) method, and Particle Swarming Optimization (PSO). Simulation results show that PSO allows to obtain accurate postion estimates with a small number of beacons, making it an attractive choice to implement effective localization algorithm.

Keywords: Particle Swarm Optimization (PSO), Auto-localization, Two-Stage Maximum-Likelihood (TSML) Algorithms, Least Square (LS) Method, Ultra Wide Band (UWB) Signaling.

1 Introduction

The problem of locating sources in an indoor environment has been widely studied since it has many applications in various areas, such as: monitoring of people in hospitals or in high security areas; search for victims or firefighters in emergency situations; home security; and locating people or vehicles in a warehouse. The use of wireless networks is an attractive option in this field, as they combine low-to-medium rate communications with positioning capabilities [6]. As a matter of fact, the distance between each pair of nodes can be estimated by sending signals between them and by extracting from these signals some physical quantities, such as the received signal strength, the angle of arrival, or the time of flight. The position of a node can then be estimated by using the distance measurements from a certain number of nodes with known positions, denoted as "beacons." The accuracy of the obtained position estimate depends on the

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 42–51, 2013.
© Springer-Verlag Berlin Heidelberg 2013

errors that affect wireless communications between nodes, which, in indoor environments, are mainly due to non-line-of-sight, multipath, and multiple access interference. To reduce the impact of these sources of errors (thus obtaining a more accurate position estimate), Ultra Wide Band (UWB) signaling is a promising technology, since, on one hand, the large bandwidth allows to penetrate through obstacles and to resolve multipath components and, on the other hand, the high time resolution improves the ranging capability [14].

In this paper, the considered scenario is a warehouse in which fixed Anchor Nodes (ANs) with known positions are used to locate Target Nodes (TNs), such as people and vehicles. A very large number of ANs might be necessary to guarantee accurate TN estimation in every accessible point inside a large building, and their accurate positioning could be very demanding also from an economic point of view. Moreover, if the geometry of the warehouse changes (e.g., for varying quantities of stored goods), the ANs might be replaced and/or new fixed ANs might be added. To overcome this problem, we focus on the auto-localization of the ANs assuming to know the exact positions of only a few beacons. The number of beacons should be small, in order to reduce the cost of installation, but sufficiently large to guarantee a reliable position estimate of other ANs. The focus of this work is to investigate the impact of the number of beacons on the system performance.

We assume to use UWB signaling. The distances between pairs of nodes are estimated by means of a time-based approach. More precisely, we consider a Time Difference Of Arrival (TDOA) approach, which is based on the estimation of the difference between the arrival times of signals traveling between each node to locate and beacons.

Many location estimate techniques, based on range measurement, have been proposed in the literature. Among them, it is worth recalling iterative methods, such as those based on Taylor series expansion [5], or the steepest-descent algorithm [9]. These techniques guarantee fast convergence for an initial value close to the true solution, which is often difficult to obtain in practice, but they are computationally expensive and convergence is not guaranteed (for instance, ignoring higher order terms in the Taylor series expansion may lead to significant errors). To overcome these limitations, closed-form algorithms have been studied, such as the Plane Intersection (PI) method [12] and the Two-Stage Maximum-Likelihood (TSML) method [2]. These methods can be re-interpreted as possible approaches to solve a minimization problem. According to this perspective, the location estimate can then be found by means of optimization techniques. More precisely, by re-formulating the initial system of equations of the TSML in terms of an optimization problem, we solve it through the use of Particle Swarming Optimization (PSO). In this work we show that the proposed approach can perform better than the PI and the TSML methods.

This paper is organized as follows. In Section 2, the PI and TSML methods and the PSO algorithm are described. In Section 3 numerical results, relative to the impact of the number of beacons on the performances of the different algorithms, are presented. Section 4 concludes the paper.

2 Scenario Description

Throughout the paper, we assume that all the ANs lay on a plane, which could be, for instance, the ceiling of a warehouse. We suppose that M beacons, whose coordinates are denoted by $\underline{s}_i = [x_i, y_i]^T, \forall i = 1, \ldots, M$ are used to get the position estimate of each AN with unknown position. In order to apply the algorithms outlined in the remainder of this section, a necessary condition is that $M \geq 4$.

If we define $\underline{u}_e = [x_e, y_e]^T$ as the true position of a generic AN (whose position needs to be estimated) and $\hat{\underline{u}}_e = [\hat{x}_e, \hat{y}_e]^T$ as its estimated position, then the true and estimated distances between the $i-$th beacon and the AN of interest are, respectively:

$$r_i = \sqrt{(\underline{u}_e - \underline{s}_i)^T(\underline{u}_e - \underline{s}_i)} \qquad \hat{r}_i = \sqrt{(\hat{\underline{u}}_e - \underline{s}_i)^T(\hat{\underline{u}}_e - \underline{s}_i)}. \tag{1}$$

Since we are considering UWB signaling, it can be shown that $\hat{r}_i \simeq r_i + \nu_i$, where $\nu_i = \varepsilon_i + b$, $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$, ε_i is independent from ε_j if $i \neq j$ $(j = 1, \ldots, M)$, and b is a synchronization bias [1]. Moreover, according to [1], the standard deviation σ_i of the position error estimation between two UWB nodes can be approximated as a linear function of the distance between them, namely

$$\sigma_i \simeq \sigma_0 r_i + \beta. \tag{2}$$

In the following, the values $\sigma_0 = 0.01$ m and $\beta = 0.08$ m are considered. These values are obtained in [1] by considering Channel Model 3 described in [10] and the energy detection receiver presented in [3], which is composed by a band-pass filter followed by a square-law device and an integrator, with integration interval set to $T_s = 1$ s. The results presented in the following hold under these channel and receiver assumptions.

In the remainder of this section, the following notation will be used:

$$\Delta_{1i} = r_i - r_1, \quad \forall\, i = 2, \ldots, M \qquad K_i = x_i^2 + y_i^2 \quad \forall\, i = 1, \ldots, M. \tag{3}$$

2.1 TSML Method

According to the TSML method, each TDOA measurement identifies a hyperbola which the source has to belong to. Therefore, given a set of TDOA measurements, the position estimate can be determined by solving the system of equations corresponding to these hyperbolas using a Least Square (LS) technique [2]. Observing that $r_i^2 = (\Delta_{1i} + r_1)^2$, from (1) and (3) the following TDOA non-linear equations can be derived:

$$\Delta_{1i}^2 + 2\Delta_{1i} r_1 = -2x_i x_e - 2y_i y_e + x_e^2 + y_e^2 - K_i - r_1^2 \qquad i = 2, \ldots, M. \tag{4}$$

When using estimated distances instead of the real ones, defining $\hat{\underline{\phi}}_1 = [\hat{\underline{u}}_e^T, \hat{r}_1]^T$, the set of equations (4) can be written as

$$\hat{\underline{\underline{G}}}\, \hat{\underline{\phi}}_1 = \hat{\underline{h}} \tag{5}$$

where

$$\hat{\underline{\underline{G}}} = -\begin{pmatrix} x_2 - x_1 & y_2 - y_1 & \hat{\Delta}_{12} \\ x_3 - x_1 & y_3 - y_1 & \hat{\Delta}_{13} \\ \vdots & \vdots & \vdots \\ x_M - x_1 & y_M - y_1 & \hat{\Delta}_{1M} \end{pmatrix} \qquad \hat{\underline{h}} = \frac{1}{2}\begin{pmatrix} K_1 - K_2 + \hat{\Delta}_{12}^2 \\ \vdots \\ K_1 - K_M + \hat{\Delta}_{1M}^2 \end{pmatrix} \tag{6}$$

and $\hat{\Delta}_{1i} = \hat{r}_i - \hat{r}_1, \forall i = 2, \ldots, M$. This is a non-linear system because, according to (1), $\hat{r}_1$ depends on $\hat{x}_e$ and $\hat{y}_e$. The solution of (5) is determined in 2 steps. First, $\hat{x}_e$, $\hat{y}_e$, and $\hat{r}_1$ are assumed to be three independent variables and the (linear) system is solved by using the LS method. Consider the error vector

$$\underline{\psi} \triangleq \hat{\underline{\underline{G}}}(\hat{\underline{\phi}}_1 - \underline{\phi}_1) \tag{7}$$

where $\underline{\phi}_1 = [x_e, y_e, r_1]^T$. The Maximum Likelihood (ML) estimate of $\hat{\underline{\phi}}_1$ is

$$\hat{\underline{\phi}}_1 = (\hat{\underline{\underline{G}}}^T \underline{\underline{\Psi}}^{-1} \hat{\underline{\underline{G}}})^{-1} \hat{\underline{\underline{G}}}^T \underline{\underline{\Psi}}^{-1} \hat{\underline{h}} \tag{8}$$

and $\underline{\underline{\Psi}} \triangleq \text{cov}(\underline{\psi}) = \underline{\underline{B}}\,\underline{\underline{Q}}\,\underline{\underline{B}}$ where $\underline{\underline{B}} = \text{diag}(r_2, \ldots, r_M)$, $\underline{\underline{Q}} = \mathbb{E}[\underline{\varepsilon}_1 \underline{\varepsilon}_1^T]$ and $(\underline{\varepsilon}_1)_j = \hat{\Delta}_{1j} - \Delta_{1j}$ [7]. It can be shown that $\text{cov}(\hat{\underline{\phi}}_1) = (\underline{\underline{G}}^T \underline{\underline{\Psi}}^{-1} \underline{\underline{G}})^{-1}$ [2]. Taking into account the relation between $\hat{x}_e$, $\hat{y}_e$, and $\hat{r}_1$, i.e., equation (1), the following set of equations can be obtained:

$$\underline{\psi}' = \hat{\underline{h}}' - \underline{\underline{G}}' \hat{\underline{\phi}}_2 \tag{9}$$

where

$$\hat{\underline{h}}' = [([\hat{\underline{\phi}}_1]_1 - x_1)^2, ([\hat{\underline{\phi}}_1]_2 - y_1)^2, [\hat{\underline{\phi}}_1]_3^2]^T \qquad \underline{\underline{G}}' = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\hat{\underline{\phi}}_2 = [(\hat{x}_e - x_1)^2, (\hat{y}_e - y_1)^2]^T.$$

The ML solution of (9) is

$$\hat{\underline{\phi}}_2 = (\underline{\underline{G}}'^T \underline{\underline{\Psi}}'^{-1} \underline{\underline{G}}')^{-1} \underline{\underline{G}}'^T \underline{\underline{\Psi}}'^{-1} \hat{\underline{h}}' \tag{10}$$

where $\underline{\underline{\Psi}}' \triangleq \text{cov}(\underline{\psi}') = 4\underline{\underline{B}}' \text{cov}(\hat{\underline{\phi}}_1) \underline{\underline{B}}'$ and $\underline{\underline{B}}' = \text{diag}(x_e - x_1, y_e - y_1, r_1)$ [7]. This leads to the following position estimate

$$\hat{\underline{u}}_e = \underline{\underline{U}} \left[\sqrt{[\hat{\underline{\phi}}_2]_1}, \sqrt{[\hat{\underline{\phi}}_2]_2} \right]^T + \underline{s}_1$$

where $\underline{\underline{U}} = \text{diag}[\text{sgn}(\hat{\underline{\phi}}_1 - \underline{s}_1)]$.

2.2 PI Method

According to the PI method, introduced in [12], any triple of ANs (which leads to a pair of TDOA measurements) identifies the major axes of a conic, a focus of which is the position of the source. Given at least two triples of ANs, the position estimate can then be determined by solving the system given by the equations of the corresponding axes. By considering the axes identified by $\{\underline{s}_1, \underline{s}_2, \underline{s}_k\}$, $k = 3, \ldots, M$ the system can be written as

$$\underline{\underline{\hat{A}}}\,\underline{\hat{u}}_e = \underline{\hat{b}} \tag{11}$$

where

$$\underline{\underline{\hat{A}}} = \begin{pmatrix} x_{21}\hat{\Delta}_{13} - x_{31}\hat{\Delta}_{12} & y_{21}\hat{\Delta}_{13} - y_{31}\hat{\Delta}_{12} \\ x_{21}\hat{\Delta}_{14} - x_{41}\hat{\Delta}_{12} & y_{21}\hat{\Delta}_{14} - y_{41}\hat{\Delta}_{12} \\ \vdots & \vdots \\ x_{21}\hat{\Delta}_{1M} - x_{M1}\hat{\Delta}_{12} & y_{21}\hat{\Delta}_{1M} - y_{M1}\hat{\Delta}_{12} \end{pmatrix} \tag{12}$$

and

$$\underline{\hat{b}} = \frac{1}{2}\begin{pmatrix} -\hat{\Delta}_{12}\hat{\Delta}_{13}(\hat{\Delta}_{13} - \hat{\Delta}_{12}) + (K_1 - K_2)\hat{\Delta}_{13} - (K_1 - K_3)\hat{\Delta}_{12} \\ -\hat{\Delta}_{12}\hat{\Delta}_{14}(\hat{\Delta}_{14} - \hat{\Delta}_{12}) + (K_1 - K_2)\hat{\Delta}_{14} - (K_1 - K_4)\hat{\Delta}_{12} \\ \vdots \\ -\hat{\Delta}_{12}\hat{\Delta}_{1M}(\hat{\Delta}_{1M} - \hat{\Delta}_{12}) + (K_1 - K_2)\hat{\Delta}_{1M} - (K_1 - K_M)\hat{\Delta}_{12} \end{pmatrix}. \tag{13}$$

where $x_{j1} \triangleq x_1 - x_j$, $y_{j1} \triangleq y_1 - y_j$, $j = 2, \ldots, M$, and K_j and $\hat{\Delta}_{1j}$ are defined in (3). The LS solution of (11) is then given by

$$\underline{\hat{u}}_e = (\underline{\underline{\hat{A}}}^T\,\underline{\underline{\hat{A}}})^{-1}\,\underline{\underline{\hat{A}}}^T\,\underline{\hat{b}}. \tag{14}$$

2.3 PSO Algorithm

The starting point for the TSML method was the system (5) in Subsection 2.1. Through simple algebraic manipulations, this system can be written as

$$\underline{\underline{B}}\,\underline{\hat{u}}_e = \underline{\hat{t}} \tag{15}$$

where

$$\underline{\underline{B}} = -2\begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_M - x_1 & y_M - y_1 \end{pmatrix} \qquad \underline{\hat{t}} = \begin{pmatrix} \hat{r}_2^2 - \hat{r}_1^2 + K_1 - K_2 \\ \hat{r}_3^2 - \hat{r}_1^2 + K_1 - K_3 \\ \vdots \\ \hat{r}_M^2 - \hat{r}_1^2 + K_1 - K_M \end{pmatrix}. \tag{16}$$

Notice that, while in (5) both the matrix $\underline{\underline{\hat{G}}}$ and the vector $\underline{\hat{h}}$ contain noisy data, in (15) the measurements affected by noise only appear in vector $\underline{\hat{t}}$, while the

matrix $\underline{\underline{B}}$ cointains known parameters. By interpreting the system (15) as an optimization problem, its solution can be expressed as follows:

$$\underline{\hat{u}}_e = \text{argmin}_{\underline{u}} ||\hat{t} - \underline{\underline{B}}\,\underline{u}||. \tag{17}$$

The PSO algorithm, introduced in [8], can be used to solve this problem. According to this algorithm, the set of potential solutions of an optimization problem is modeled as a swarm of S particles, which are guided towards the optimal solution of the given problem, by exploiting "social" interactions between individuals [11]. It is assumed that every particle i in the swarm ($i = 1, \ldots, S$) at any given instant t is associated with a position $\underline{x}^{(i)}(t)$ in the region of interest and with a velocity $\underline{v}^{(i)}(t)$, which are both randomly initialized at the beginning with values $\underline{x}^{(i)}(0)$ and $\underline{v}^{(i)}(0)$ and which are updated at each iteration [4]. It is also assumed that the system has memory, so that, at every instant, each particle knows not only its own best position reached so far, but also the best position among the ones reached by any other particle in the swarm in the previous iterations. Each particle also keeps track of the values of the function to optimize in correspondence to both its best position and the global best position. These values are used to update the velocity and the position of every particle at each iteration. More precisely, the velocity of particle i is updated at consecutive iterations, according to the rule [13]

$$\underline{v}^{(i)}(t+1)=\omega(t)\underline{v}^{(i)}(t)+c_1R_1(t)(\underline{y}^{(i)}(t)-\underline{x}^{(i)}(t))+c_2R_2(t)(\underline{y}(t)-\underline{x}^{(i)}(t)) \quad i = 1, \ldots, S \tag{18}$$

where: $\omega(t)$ is denoted as *inertial factor*; c_1 and c_2 are positive real parameters denoted as *cognition* and *social* parameters, respectively; $R_1(t)$ and $R_2(t)$ are random variables uniformly distributed in $(0, 1)$; and $\underline{y}^{(i)}(t)$ and $\underline{y}(t)$ are the position of the $i-$th particle with the best objective function and the position of the particle with the best (among all particles) objective function reached until instant t [11]. In the considered minimization problem (17), they can be described as

$$\begin{aligned} \underline{y}^{(i)}(t) &= \text{argmin}_{\underline{z}\in\{\underline{x}^{(i)}(0),\ldots\,\underline{x}^{(i)}(t)\}} ||\hat{t} - \underline{\underline{B}}\,\underline{z}|| \\ \underline{y}(t) &= \text{argmin}_{\underline{z}\in\{\underline{y}^{(1)}(t),\ldots\,\underline{y}^{(S)}(t)\}} ||\hat{t} - \underline{\underline{B}}\,\underline{z}||. \end{aligned} \tag{19}$$

The idea behind the iterative step (18) is to add to the previous velocity of particle i (which is weighted by means of a multiplicative factor) a stochastic combination of the direction to its best position and to the best global position. The definition of the velocity given in (18) is then used to update the position of the $i-$th particle, according to the following rule:

$$\underline{x}^{(i)}(t+1) = \underline{x}^{(i)}(t) + \underline{v}^{(i)}(t) \qquad i = 1, \ldots, S.$$

Possible stopping conditions for the PSO algorithm can be the achievement of a satisfying value of the function to be minimized or a given (maximum) number of iterations. At the end of the algorithm, the solution is the position of the particle which best suits the optimization requirements in the last iteration.

The application of PSO to the considered localization problem is better explained in the next section.

3 Simulation Results

In this section, the three localization approaches described in Section 2, namely TSML, PI, and PSO, are compared through MATLAB based simulations compliant with the propagation model introduced in Section 2. In all cases, the performance is evaluated in terms of Mean Square Error (MSE) between true and estimated positions, i.e.:

$$\text{MSE} \triangleq \mathbb{E}[(\hat{x}_e - x_e)^2 + (\hat{y}_e - y_e)^2]. \tag{20}$$

In the following simulations, the MSE is obtained from the average of 100 independent runs.

The PSO algorithm has been implemented by setting both the parameters c_1 and c_2 in (18) to 2. This choice makes the weights for social and cognition parts to be, on average, equal to 1. The inertial factor $\omega(t)$ has been chosen to be a decreasing function of the number of iterations, in order to guarantee low dependence of the solution on the initial population and to reduce the exploitation ability of the algorithm, making the method more simliar to a local search, as the number of iterations increases [13]. In the following, it is assumed that the initial value of the inertial factor is $\omega(0) = 0.9$ and that it decreases linearly to 0.4, reached at the 50-th iteration, i.e. the last one according to the stopping criterion we chose. A population of 40 particles is considered since previous simulations showed that this value is large enough to guarantee an accurate solution and that incrementing it does not lead to significant improvements.

We investigate through simulation the minimum number of beacons that are needed to obtain a reliable estimate of the ANs positions. First, we consider a partition of the entire plane on which the ANs lay into squares, whose edges are 10 m long. Without loss of generality, we restrict our analysis to a single square. The considered scenario is shown in Fig. 1 (b) and Fig. 1 (d), where circles represent beacons while squares represent ANs with unknown positions. In the scenario shown in Fig. 1 (b), 8 out of the total 36 ANs are assumed to be beacons and their known coordinates are then used to get the position estimate of the remaining 28 ANs. In Fig. 1 (a), the MSE corresponding to each AN is represented. In this case, by comparing the MSE of the three algorithm, it can be noticed that there are no significant differences in the order of magnitude of the error and the three algorithms guarantee an accurate position estimate of all the ANs. On the other hand, in the scenario represented in Fig. 1 (d) only 4 beacons are assumed to be used to estimate the positions of the remaining 32 ANs. As can be noticed from Fig. 1 (c), in this case both the TSML and the PI methods lead to a far inaccurate positioning estimate for many ANs while the accuracy obtained when using the PSO algorithm is still good. Moreover, by comparing the behaviour of PSO algorithm when 8 beacons are used with the one obtained with only 4 beacons shows that the MSE has the same order of magnitude in both cases. It can then be concluded that 4 beacons are not enough to obtain a reliable estimate when using the TSML and the PI methods, but they are sufficient to guarantee an accurate postion estimate when the PSO algorithm is used.

We now consider a scenario composed by a corridor 40 m long and 5 m wide, as shown in Fig. 2 (b) and Fig. 2 (d). In the scenario represented in Fig. 2 (b), there are 16 beacons out of the total 44 ANs and this allows to obtain an accurate position estimate with all the three approaches previously described, namely the TSML and the PI method and the PSO algorithm, as shown in Fig. 2 (a).

As in the previous scenario, reducing the number of beacons, as in Fig. 2 (d), leads to significantly worse values of the MSE corresponding both to TSML and PI method, without changing the accuracy of the position estimate obtained via the PSO algorithm, as shown in Fig. 2 (c).

Therefore, we can observe that the PSO algorithm can be successfully applied with at least half the number of beacons, which allows to save money and time in the accurate positioning of fixed ANs.

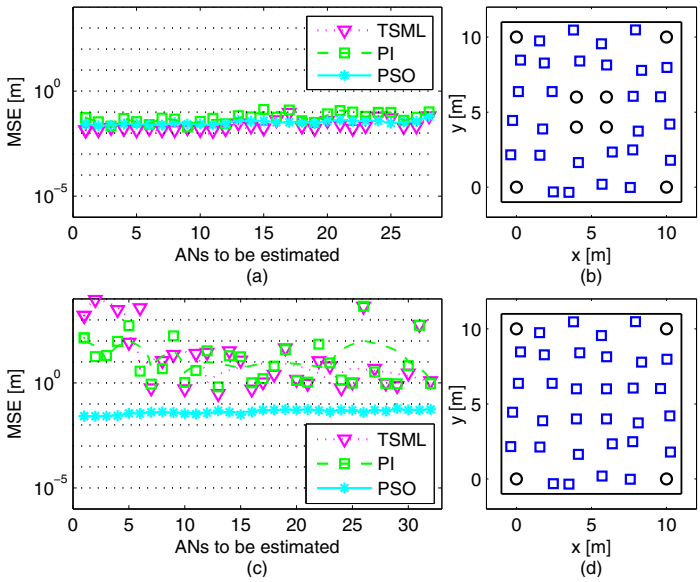


Fig. 1. Fig. 1 (b) and Fig. 1 (d) represent the beacons (*circles*) and the ANs whose positions need to be estimated (*squares*). In Fig. 1 (a.) the MSE of the ANs relative to the scenario described in Fig. 1 (b.) is plotted, corresponding to TSML method (*triangles*), PI method (*squares*) and PSO algorithm (*dots*). In each case, the interpolation lines (*dotted* lines for TSML, *dashed* lines for PI, *solid* lines for PSO) are shown. Fig. 1 (c.) represents the MSE realtive to each AN when the considered scenario is the one described in Fig. 1 (d.). In this case the PSO algorithm outperforms the TSML and the PI method, showing that 4 beacons are enough to obtain an accurate position estimate when using PSO.

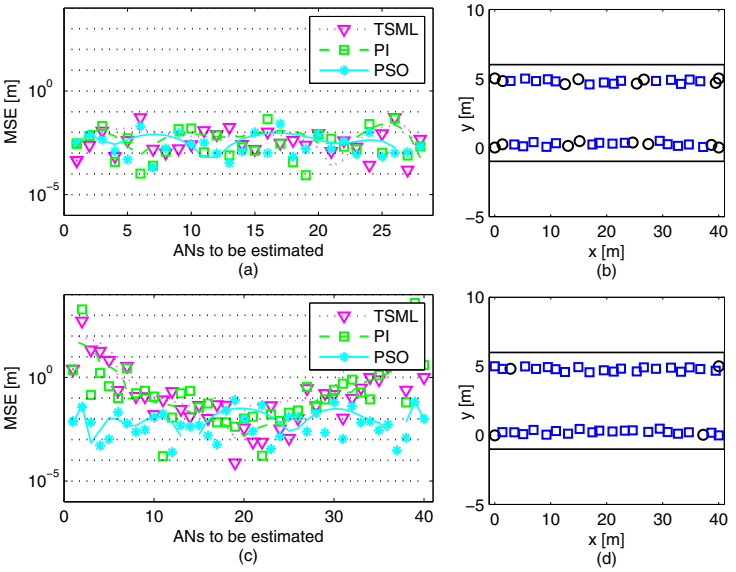


Fig. 2. Fig. 2 (b) and Fig. 2 (d) represent the beacons (*circles*) and the ANs whose positions need to be estimated (*squares*). In Fig. 2 (a) the estimation MSEs of the ANs in the scenario described in Fig. 2 (b) are shown, using TSML method (*triangles*), PI method (*squares*) and PSO algorithm (*dots*). In this case the order of magnitude of the error is the same for all the three algorithms. Fig. 2 (c) represents the MSE realtive to each AN when the considered scenario is the one described in Fig. 2 (d). In this case the PSO algorithm outperforms the TSML and the PI method, showing that 4 beacons are enough to obtain an accurate position estimate when using PSO.

4 Conclusion

In order to evaluate the impact of the number of beacons on localization accuracy, three approaches to UWB-signaling-based auto-localization of nodes in a static wireless network have been considered. Besides solving the localization problem by means of the TSML and the PI methods, which are widely used for this purpose, the original system of non-linear equations of the TSML method has been re-formulated in terms of an optimization problem, which is then solved by means of PSO. Our results show that the PSO approach guarantees a good accuracy in the position estimate with a smaller number of known beacons, allowing to reduce the installation cost of the entire localization system.

Acknowledgment. This work is supported by Elettric80 (http://www.elettric80.it).

References

1. Busanelli, S., Ferrari, G.: Improved ultra wideband-based tracking of twin-receiver automated guided vehicles. Integrated Computer-Aided Engineering 19(1), 3–22 (2012)
2. Chan, Y., Ho, K.C.: A simple and efficient estimator for hyperbolic location. IEEE Trans. Signal Process. 42(8), 1905–1915 (1994)
3. Dardari, D., Chong, C.C., Win, M.Z.: Threshold-based time-of-arrival estimators in uwb dense multipath channels. IEEE Trans. Commun. 56(8), 1366–1378 (2008)
4. Eberhart, R., Kermedy, J.: A new optimizer using particles swarm theory. In: Proc. Sixth International Symposium on Micro Machine and Hmm Science, Nagoya, Japan. IEEE Service Center, Piscataway (1995)
5. Foy, W.H.: Position-location solutions by Taylor-series estimation. IEEE Trans. Aerosp. Electron. Syst. AES-12(2), 187–194 (1976)
6. Gezici, S., Poor, H.V.: Position estimation via ultra-wide- band signals. Proc. IEEE 97(2), 386–403 (2009)
7. Ho, K.C., Xu, W.: An accurate algebraic solution for moving source location using TDOA and FDOA measurements. IEEE Trans. Signal Process. 52(9), 2453–2463 (2004)
8. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE International Conf. on Neural Networks, Perth, Australia. IEEE Service Center, Piscataway (1995)
9. Mensing, C., Plass, S.: Positioning algorithms for cellular networks using TDOA. In: 2006 IEEE International Conference on Proceedings of the Acoustics, Speech and Signal Processing, ICASSP 2006, vol. 4 (May 2006)
10. Molisch, A.F., Cassioli, D., Chong, C.-C., Emami, S., Fort, A., Kannan, B., Karedal, J., Kunisch, J., Schantz, H.G., Siwiak, K., Win, M.Z.: A comprehensive standardized model for ultrawideband propagation channels. IEEE Trans. Antennas Propagat. 54(11), 3151–3166 (2006)
11. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence Journal 1(1) (2007)
12. Schmidt, R.O.: A new approach to geometry of range difference location. IEEE Trans. Aerosp. Electron. Syst. AES-8(6), 821–835 (1972)
13. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proc. IEEE International Conference on Evolutionary Computation, Piscataway, NJ, pp. 69–73 (1999)
14. Zhang, J., Orlik, P.V., Sahinoglu, Z., Molisch, A.F., Kinney, P.: UWB systems for wireless sensor networks. Proc. IEEE 97(2), 313–331 (2009)

Load Balancing in Distributed Applications Based on Extremal Optimization

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

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

Abstract. The paper shows how to use Extremal Optimization in load balancing of distributed applications executed in clusters of multicore processors interconnected by a message passing network. Composed of iterative optimization phases which improve program task placement on processors, the proposed load balancing method discovers dynamically the candidates for migration with the use of an Extremal Optimization algorithm and a special quality model which takes into account the computation and communication parameters of the constituent parallel tasks. Assessed by experiments with simulated load balancing of distributed program graphs, a comparison of the proposed Extremal Optimization approach against a deterministic approach based on a similar load balancing theoretical model is provided.

Keywords: distributed program design, extremal optimization, load balancing.

1 Introduction

Efficient execution of irregular distributed applications usually requires some kind of processor load balancing. This is even more true in a multi-user environment where, following variations in system resources availability and/or changes of their computational load, the balance in using the executive resources can notably vary over time. Thus, a dynamic load balancing facility, embedded in the runtime environment or in the distributed application is essential.

The contribution of this paper is a dynamic load balancing method based on the program and runtime system behavior observation supported by the Extremal Optimization (EO) approach [2]. The presented approach leverages some our earlier works on load balancing, reported in [3,9], and our experience in using Extremal Optimization to static task scheduling in distributed programs [4, 8]. The proposed load balancing algorithm is composed of iterative optimization phases which improve program task placement on processors to determine the

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 52–61, 2013.
© Springer-Verlag Berlin Heidelberg 2013

possibly best balance of computational loads and to define periodic migration of tasks. The Extremal Optimization is used in iterative load balancing phases which are executed in the background, in parallel with the application program. The Extremal Optimization algorithm discovers the candidate tasks for migration based on a special quality model including the computation and communication parameters of parallel tasks. The algorithm is assessed by experiments with simulated load balancing of distributed program graphs. In particular, the experiments compare the proposed load balancing method including the Extremal Optimization with an equivalent deterministic algorithm based on the similar theoretical foundations. The comparison shows that the quality of load balancing with Extremal Optimization is in most cases better than that of the deterministic algorithm.

A good review and classification of load balancing methods is presented in [1]. When we compare the proposed load balancing method to the current parallel computing environments with load balancing like CHARM++ [7] or Zoltan [5], we notice that none of them includes Extremal Optimization as a load balancing algorithm component. So, the proposed approach has clear originality features and enables making profit of the Extremal Optimization advantages such as low computational complexity and limited use of memory space.

The proposed load balancing algorithms is meant for implementation in a novel distributed program design framework which supports a programmer with an API and GUI for the automated design of program execution control based on global application states monitoring. This framework called PEGASUS (from Program Execution Governed by Asynchronous SUpervision of States) [11] provides high-level distributed control primitives at process level and a special control infrastructure for global asynchronous execution control at thread level. Monitoring of global application states in PEGASUS creates a flexible basis for the load balancing implementation as shown in [3].

The paper consists of three parts. In the first part the Extremal Optimization principles are shortly explained. The second part describes the theoretical basement for the discussed algorithm and explains how the Extremal Optimization is applied to the proposed method of dynamic load balancing. The third part explains the performed experimental assessment of the proposed algorithms.

2 Extremal Optimization Algorithm Principles

Extremal Optimization is an important nature inspired optimization method. It was proposed by Boettcher and Percus [2]. It represents a method for NP–hard combinatorial and physical optimization problems.

EO works with a single solution S consisting 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 local fitness value ϕ_i is assigned to each of them. Then, at each time step the global fitness $\Phi(S)$ is computed and S is evolved, by randomly updating only the worst variable, to a solution

S' belonging to its neighborhood $Neigh(S)$. An obtained solution is registered if its global fitness function is better than that of the best solution found so far.

To avoid sticking in a local optimum, we use a probabilistic version of EO based on a parameter τ , i.e., τ–EO, introduced by Boettcher and Percus. According to it, for a minimization problem, the species are first ranked in increasing order of local 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: $p_k \sim k^{-\tau}$, $1 \leq k \leq n$ for a given value of the parameter τ. Finally, at each update, a generic rank k is selected according to p_k so that the species s_i with $i = \pi(k)$ randomly changes its state and the solution moves to a neighboring one, $S' \in Neigh(S)$, unconditionally. The only parameters are the total number of iterations $\mathcal{N}_{\text{iter}}$ and the probabilistic selection parameter τ. For minimization problems τ–EO proceeds as in the Algorithm 1.

3 Extremal Optimization Applied to Load Balancing

3.1 System and Program Model

An *executive system* consists of N computing nodes interconnected by a message passing network (e.g. a cluster of workstations). Each node, identified by an integer value in the range $[0, N-1]$, is a multicore processor.

Distributed application programs are composed of processes and threads inside each process. The application model is similar to the model presented in [10] (Temporal Flow Graph, TFG). The application consists of the set T of indivisible tasks (these are threads in processes). Each task consists of several computational instruction blocks, separated by communication with other tasks.

The *target problem* is defined as follows: assign each task $t_k, k \in \{1 \ldots |T|\}$ of the program to a computational node $n, n \in [0, N-1]$ in such a way that the

Algorithm 1. General EO algorithm

```
initialize configuration S at will
S_best ← S
while total number of iterations N_iter not reached do
   evaluate φ_i for each variable s_i of the current solution S
   rank the variables s_i based on their fitness φ_i
   choose the rank k according to k^{-τ} so that the variable s_j with j = π(k) is selected
   choose S' ∈ Neigh(S) such that s_j must change
   accept S ← S' unconditionally
   if Φ(S) < Φ(S_best) then
      S_best ← S
   end if
end while
return S_best and Φ(S_best)
```

total program execution time is minimized, assuming the program and system representation as described earlier in this section.

The load balancing approach, proposed in the paper, consists of two main steps: the **detection** of the imbalance and its **correction**. The first step uses some measurement infrastructure to detect the functional state of the computing system and executed application. In parallel with the execution of an application, computing nodes periodically report their loads to the load balancing controller which evaluates the current system load imbalance value. Depending on this value, the second step (i.e. the imbalance correction) is undertaken or the step one is repeated. In the second step, we execute the EO-based algorithm described in next sections, which determines the set of tasks for migration and the migration target nodes. Then we perform the physical task migrations following the best solution found by the EO algorithm and go to step one.

The *state of the system* is expressed in the terms of:

$Ind_{power}(n)$ — computing power of a node n, which is the sum of computing powers of all cores on the node,
$Time^{\%}_{\mathrm{CPU}}(n)$ — the percentage of the CPU power available for computing threads on the node n, periodically estimated by observation agents on computing nodes.
The state metrics are used to detect and correct a load imbalance between nodes.

3.2 Detection of Load Imbalance

Computing nodes in a parallel system can be heterogeneous, therefore to detect the current load imbalance in the system, we will base it on the percentage of the CPU power available $Time^{\%}_{\mathrm{CPU}}(n)$ for computing threads on the nodes.

A current load imbalance LI is defined based on the difference of the CPU availability between the most heavily and the least heavily loaded computing nodes composing the cluster in the following way:

$$LI = \begin{cases} \mathit{true} \text{ if } \max_{n \in N}(Time^{\%}_{\mathrm{CPU}}(n)) - \min_{n \in N}(Time^{\%}_{\mathrm{CPU}}(n)) \geq \alpha \\ \mathit{false} \text{ otherwise} \end{cases}$$

where: N — the set of all computing nodes. The current detected load imbalance requires a load balancing correction. The value of the α is set using a statistical or/and experimental approach. Following our previous research [9] on load balancing algorithms for Java-based distributed environment, this value should be between 25% and 75%.

When $LI = true$, then, according to the load of each node, the set of computing nodes is divided into three categories, based on the computed power availability indexes: overloaded, normally loaded and underloaded. To build categories, we use the K-Means algorithm [6] with $K = 3$. The three centers that we choose are the minimum, average and maximum availability indexes, where the average index is simply the average of indexes measured during the last series of measures over the whole cluster.

3.3 Load Balancing Procedure

Now, we are able to perform the second step of our approach leading to migration of application tasks to balance the load of the system.

The **application** is characterized by two metrics, which should be provided by a programmer based on the volume of computation and communication introduced by application tasks:

1. $\mathrm{COM}(t_s, t_d)$ is the communication metrics between tasks t_s and t_d,
2. $\mathrm{WP}(t)$ is the load weight metrics introduced by a task t.

A **mapping solution** S is represented by a vector $\mu = (\mu_1, \ldots, \mu_{|T|})$ of $|T|$ integers ranging in the interval $[0, N-1]$, where the value $\mu_i = j$ means that the solution S under consideration maps the i–th task t_i of the application onto computing node j. The number of processor cores is not represented inside the solution encoding, however, it is implicitly taken into account (via the $Ind_{power}(n)$ function) when estimating the global and local fitness functions while solving the scheduling problem.

The **global fitness** function $\Phi(S)$ is defined as follows.

$$\Phi(S) = attrExtTotal(S) * \Delta_1 + migration(S) * \Delta_2 + imbalance(S) * [1 - (\Delta_1 + \Delta_2)]$$

where $1 > \Delta_1 \geq 0$, $1 > \Delta_2 \geq 0$ and $\Delta_1 + \Delta_2 < 1$ hold.

The function $attrExtTotal(S)$ represents the total external communication between tasks for given mapping S. By "external" we mean the communication between tasks placed on different nodes (i.e. which have to be transmitted actually through communication links between computing nodes). The value of this function is normalized in the range $[0, 1]$, i.e. it is a quotient of an absolute value of the total external communication volume and the total communication volume of all communications (when all tasks are placed on the same node $attrExtTotal(S) = 0$, when tasks are placed in the way that all communication became external $attrExtTotal(S) = 1$):

$$attrExtTotal(S) = totalExt(S)/\overline{\mathrm{COM}}$$

where: $\overline{\mathrm{COM}} = \sum_{s,d \in T} \mathrm{COM}(s,d)$ and $totalExt(S) = \sum_{s,d \in T: \mu_s \neq \mu_d} \mathrm{COM}(s,d)$.

The function $migration(S)$ is a migration costs metrics. The value of this function is in the range $[0, 1]$, i.e. it is equal to 0 when there is no migration, when all tasks have to be migrated $migration(S) = 1$, otherwise $0 \leq migration(S) \leq 1$:

$$migration(S) = |\{t \in T : \mu_t^S \neq \mu_t^{S*}\}|/|T|$$

where: S is the currently considered solution and $S*$ is the previous solution (or the initial solution at the start of the algorithm).

The function $imbalance(S)$ represents the numerical load imbalance metrics in the solution S. It is equal to 1 when in S there exists at least one unloaded (empty) computing node, otherwise it is equal to the normalized average absolute load deviation of tasks in S, determined in the definition below:

$$imbalance(S) = \begin{cases} 1 & \text{exists at least one unloaded node} \\ D^*(S)/2 * N * \overline{\mathrm{WP}} & \text{otherwise} \end{cases}$$

where: $D^*(S) = \sum_{n\in[0,N-1]} |\mathrm{NWP}(S,n)/Ind_{power}(n) - \overline{\mathrm{WP}}|$,
$\mathrm{NWP}(S,n) = \sum_{t\in T:\mu_t=n} \mathrm{WP}(t)$, $\overline{\mathrm{WP}} = \sum_{t\in T} \mathrm{WP}(t)/\sum_{n\in[0,N-1]} Ind_{power}(n)$.

In the applied EO the **local fitness** function of a task $\phi(t)$ is designed in such a way that it forces moving tasks away from overloaded nodes, at the same time preserving low external (inter-node) communication. The γ parameter ($0 < \gamma < 1$) allows tuning the weight of load metrics.

$$\phi(t) = \gamma * load(\mu_t) + (1-\gamma) * rank(t)$$

The function $load(n)$ indicates whether the node n, which executes t is overloaded (i.e. it indicates how much its load exceeds the average load of all nodes):

$$load(n) = \frac{load^*(n)}{\max_{m\in[0,N-1]} load^*(m)}, \quad load^*(n) = \max(\frac{\mathrm{NWP}(S,n)}{Ind_{power}(n)} - \overline{\mathrm{WP}}, 0).$$

The $rank(t)$ function governs the selection of best candidates for migration. The chance for migration have tasks, which have low communication with their current node (attraction) and low load deviation from the average load:

$$rank(t) = 1 - (\beta * attr(t) + (1-\beta) * ldev(t))$$

where: β is a real number between 0 and 1 — a parameter indicating the importance of the weight of attraction metrics.

The attraction of the task t to its executive computing node is defined as:

$$attr(t) = \frac{attr^*(t)}{\max_{o\in L(t)}(attr^*(o))}$$

where: $attr^*(t) = \sum_{o\in L^*(t)}(\mathrm{COM}(t,o) + \mathrm{COM}(o,t))$ — the amount of communication between task t and other tasks on the same node, $L(t) = \{u \in T : \mu_t = \mu_u\}$ — the set of threads, placed on the same node as the thread t (including t), $L^*(t) = \{u \in T, u \neq t : \mu_t = \mu_u\}$ — the set of threads, placed on the same node as a thread t (excluding t).

The load deviation compared to the average load is defined as:

$$ldev(t) = \frac{ldev^*(t)}{\max_{o\in L(t)}(ldev^*(o))}$$

where: $ldev^*(t) = |\mathrm{WP}(t) - mean_{\mathrm{WP}}(t)|$, $mean_{\mathrm{WP}}(t) = \sum_{o\in L(t)} \mathrm{WP}(o)/|L(t)|$.

3.4 Deterministic Approach for Load Balancing

For comparison purposes, we have implemented also a fully deterministic load balancing algorithm, based on similar migration criteria as shown in the previous section (see [3] for the description). Similarly to the EO approach, the load balancing triggering is controlled by the imbalance metrics LI. The deterministic algorithm iterates over all overloaded nodes and migrates a single task from each such node to an underloaded one. The task for migration is selected according to the $rank(t)$ function. The target of migration is the node that minimizes the weighted sum of $attrExtTotal(S)$ and $Time^{\%}_{\mathrm{CPU}}(n)$.

4 Experimental Assessment of Load Balancing Algorithms

We will present now an experimental assessment of the presented load balancing algorithm. The experimental results have been obtained by simulated execution of application programs in a distributed system. The assumed simulated model of execution corresponds to parallelization based on message-passing, using the MPI library for communication. The applied simulator was built following the DEVS discrete event system approach [12].

Applications were run in a cluster of multi-core processors, each of which had its own main memory and a network interface. Communication contention was modeled at the level of the network interface of each computing node.

During experiments we used a set of 10 synthetic exemplary programs, modeled as TFG (see section 3.1). These programs were randomly generated, but their general structure resembled typical MPI-based parallel applications which correspond to numerical analysis or physical phenomena simulation. Each application program consisted of a set of program modules, Fig 1. A module was composed of parallel tasks (threads). Tasks of the same module communicated. At the boundaries between modules, there was a global exchange of data.

The number of tasks in an application varies from 16 to 80. The communication/computation ratio (the quotient of the communication time to the execution time in our experimental environment) for applications is in the range 0.05...0.15. Three applications have regular tasks' execution times. The difference between regular and irregular applications is that the execution time of tasks in the irregular applications depends on the processed data. In regular applications a load imbalance can appear due to the placement of multiple tasks on the same processor.

In the first experiment, we simulated execution of applications in systems with 2, 3, 4 and 8 identical computing nodes. We used the following parameters: $\alpha = 0.5, \beta = 0.5, \gamma = 0.5, \Delta_1 = 0.25, \Delta_2 = 0.25, \tau = 1.5$. The number of iterations of the EO algorithm was set to 500. The results are the averages of 5 runs of each application, each run for 4 different methods of initial task placements (random, round-robin, METIS, packed) i.e. 20 runs for each parameter set.

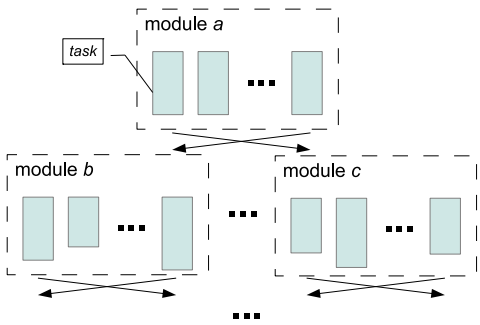


Fig. 1. The general structure of exemplary applications

Table 1. Speed-up improvement for irregular and regular applications due to load balancing for different number of nodes (2,3,4,8) in the system

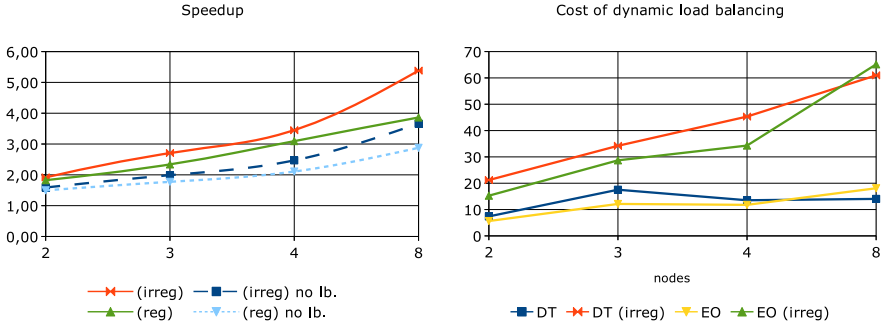
algorithm		2	3	4	8	average
EO – extremal optimization	irregular	18.66%	33.42%	37.32%	41.91%	35.08%
	regular	21.95%	31.85%	46.85%	34.21%	34.71%
DT – deterministic	irregular	18.27%	28.17%	33.97%	43.82%	33.80%
	regular	22.95%	29.54%	41.62%	35.02%	33.39%

The speed-up improvement resulting from load balancing performed by the EO-based algorithm and the deterministic approach (DT) is shown in Tab. 1. The general speed-up improvement over the execution without load balancing is bigger for EO-based algorithm. As we'll show later in this section, it is possible to obtain even better speedup by EO through proper parameter tuning.

In Fig. 2(a) the speed-up of irregular and regular applications for different number of computing nodes is shown. Our exemplary regular applications give smaller speed-up than irregular ones (with or without load balancing).

Since migration costs can be very different (the single migration can be as short as a simple task activation message, but also it can involve a transfer of the processed data, which is usually very costly), we decided to measure also the imposed load balancing costs as the number of tasks migrations. As shown in Fig. 2(b), the average cost imposed by EO algorithm is generally lower than the cost introduced by the deterministic approach.

The big advantage of the EO approach is the ability to tune its behavior through the algorithm parameters setting. Thus, we can have bigger speedup at a higher cost of migration, or lower speedup with a reduced migrations number. To do so, we performed the simulations for different values of $\tau \in \{0.75, 1.5, 3.0\}$ and varying sets of $\gamma, \Delta_1, \Delta_2$ factors. We considered the following combinations of $\gamma, \Delta_1, \Delta_2$: **U05** (imbalance least important): $\gamma = 0.5, \Delta_1 = 0.25, \Delta_2 = 0.25$;



(a) (b)

Fig. 2. (a) Speedup for different number of nodes with and without load balancing (b) Cost of the dynamic load balancing as the number of task migrations per single execution of an application

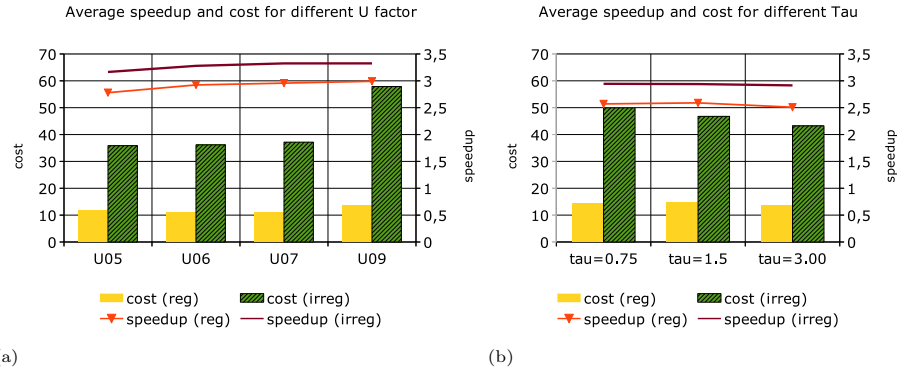


Fig. 3. Speedup and cost for different values of EO parameters as a function of: (a) U0x, (b) τ (tau)

U06: $\gamma = 0.6, \Delta_1 = 0.18, \Delta_2 = 0.22$; **U07**: $\gamma = 0.75, \Delta_1 = 0.13, \Delta_2 = 0.17$; **U09** (imbalance most important): $\gamma = 0.95, \Delta_1 = 0.05, \Delta_2 = 0.05$.

For the tested applications, better results were obtained when the load imbalance was the primary optimization factor, namely for U07 and U09, Fig. 3(a). On the other hand, for U09 the cost of load balancing was very high (the algorithm migrates tasks very often to maintain a perfect load balance). We expect that for applications with more intense communication (i.e. higher value of communication/computation ratio) better results can be obtained for U05 and U06. It will be investigated in further research.

Fig. 3(b) shows that an increasing value of τ decreases the number of migrations, it reduces also slightly the obtained speedup (note that increasing τ increases the probability of selection of the worst species for change).

Our experimental results collected so far by simulation confirm that the presented EO-based load balancing method performs well for different run-time conditions. Other important features of the method are the low migration overhead as well as the ease of programming and tuning the load balancing algorithm.

5 Conclusions

Dynamic load balancing in distributed systems based on application of the Extremal Optimization approach and global states monitoring has been discussed in this paper. The load balancing algorithm composed of iterative optimization phases based on the use of Extremal Optimization which define periodic real migration of the tasks proved to be an efficient and successful method for load balancing. The Extremal Optimization is executed in the background of application computations, in parallel with the application program.

The proposed algorithm including the Extremal Optimization has been assessed by experiments with simulated load balancing of distributed program graphs. In particular, the experiments compare the proposed load balancing

method including the Extremal Optimization with an equivalent deterministic algorithm based on the similar theoretical foundations for load balancing. The comparison shows that the quality of load balancing with Extremal Optimization is in most cases better than that of the deterministic algorithm. The proposed load balancing method is meant for implementation in the PEGASUS distributed program design framework, which is in the final implementation stage.

Acknowledgments. This paper has been partially sponsored by the MNiSW grant No. NN 516 367 536.

References

1. Barker, K., Chrisochoides, N.: An Evaluation of a Framework for the Dynamic Load Balancing of Highly Adaptive and Irregular Parallel Applications. In: Supercomputing 2003. ACM, Phoenix (2003)
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. Borkowski, J., Kopański, D., Laskowski, E., Olejnik, R., Tudruj, M.: A Distributed Program Global Execution Control Environment Applied to Load balancing. Scalable Computing: Practice and Experience 13(3) (2012)
4. 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, Part I. LNCS, vol. 6271, pp. 180–191. Springer, Heidelberg (2010)
5. Devine, K.D., Boman, E.G., Riesen, L.A., Catalyurek, U.V., Chevalier, C.: Getting Started with Zoltan: A Short Tutorial, Sandia National Labs Tech Report SAND2009-0578C (2009)
6. Hartigan, J.A., Wong, M.A.: A K-Means clustering algorithm. Applied Statistics 28, 100–108 (1979)
7. Kalé, L.V., Krishnan, S.: CHARM++: A Portable Concurrent Object Oriented System Based on C++. In: Proc. of OOPSLA 1993. ACM Press (September 1993)
8. Laskowski, E., Tudruj, M., De Falco, I., Scafuri, U., Tarantino, E., Olejnik, R.: Extremal Optimization Applied to Task Scheduling of Distributed Java Programs. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Drechsler, R., Farooq, M., Grahl, J., Greenfield, G., Prins, C., Romero, J., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Urquhart, N., Uyar, A.Ş. (eds.) EvoApplications 2011, Part II. LNCS, vol. 6625, pp. 61–70. Springer, Heidelberg (2011)
9. Olejnik, R., Alshabani, I., Toursel, B., Laskowski, E., Tudruj, M.: Load Balancing Metrics for the SOAJA Framework. Scalable Computing: Practice and Experience 10(4) (2009)
10. Roig, C., Ripoll, A., Guirado, F.: A New Task Graph Model for Mapping Message Passing Applications. IEEE Trans. on Parallel and Distributed Systems 18(12), 1740–1753 (2007)
11. Tudruj, M., Borkowski, J., Maśko, Ł., Smyk, A., Kopański, D., Laskowski, E.: Program Design Environment for Multicore Processor Systems with Program Execution Controlled by Global States Monitoring. In: ISPDC 2011, Cluj-Napoca, pp. 102–109. IEEE CS (July 2011)
12. Zeigler, B.: Hierarchical, modular discrete-event modelling in an object-oriented environment. Simulation 49(5), 219–230 (1987)

A Framework for Modeling Automatic Offloading of Mobile Applications Using Genetic Programming

G. Folino and F.S. Pisani

Institute of High Performance Computing and Networking (ICAR-CNR), Italy
{folino,fspisani}@icar.cnr.it

Abstract. The limited battery life of the modern mobile devices is one of the key problems limiting their usage. The offloading of computation on cloud computing platforms can considerably extend the battery duration. However, it is really hard not only to evaluate the cases in which the offloading guarantees real advantages on the basis of the requirements of application in terms of data transfer, computing power needed, etc., but also to evaluate if user requirements (i.e. the costs of using the clouds, a determined QoS required, etc.) are satisfied. To this aim, in this work it is presented a framework for generating models for taking automatic decisions on the offloading of mobile applications using a genetic programming (GP) approach. The GP system is designed using a taxonomy of the properties useful to the offloading process concerning the user, the network, the data and the application. Finally, the fitness function adopted permits to give different weights to the four categories considered during the process of building the model.

1 Introduction

The introduction of larger screens and the large usage and availability of cpu-consuming and network-based mobile applications further reduce the battery life of mobile devices. Therefore, due to these problems and to the proliferations of mobile devices (i.e. tablets and smartphones), the interest in trying to improve the limited life of their batteries is greatly increased. A possible solution to alleviate this problem is to offload part of the application or the whole computation to remote servers, as explained in [6], where software-based techniques for reducing program power consumption are analyzed, considering both static and dynamic information in order to move the computation to remote servers.

In the last few years, the emergence of the Cloud Computing technology and the consequent large availability of cloud servers [2], encouraged the research into the usage of offloading techniques on cloud computing platforms. A number of papers were published trying to cope with the main issues of the process of offloading, mainly oriented toward a particular problematic, i.e. wifi issues [8], network behavior [12] and network bandwidth [15], the tradeoff between privacy and quality [9] and the effect of the context [1].

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 62–71, 2013.
© Springer-Verlag Berlin Heidelberg 2013

However, to the best of our knowledge, the works concerning this thematic do not report a complete model keeping into account both the hardware/software issues and the user's requirements, neither an automatic and adaptive model to take the decision of performing the offloading. Indeed, the offloading technique potentially could improve both performance and energy consumption, however it is a NP-hard problem to establish whether it is convenient to perform the migration, especially considering all the correlated problems such as network disconnections and variability, privacy and security of the data, variations of load in the server, etc. In this work, we try to overcome these limitations, by designing a framework for modeling automatic offloading of mobile application using a genetic programming approach.

Logically, we divided the software architecture of the framework into two parts: a module for simulating the offloading process and an inference engine for building an automatic decision model on the same process of offloading. Both of them are based on a taxonomy that defines four main categories concerning the offloading process: user, network, data and application.

The simulator evaluates the performance of the offloading process of mobile applications on the basis of the user requirements, of the conditions of the network, of the hardware/software features of the mobile device and of the characteristics of the application. The inference engine is used to generate decision trees able to model the decisions on the offloading process on the basis of the parameters contained in the categories defined by the taxonomy. It is constituted by a genetic programming environment, in which the functions are naturally defined by the taxonomy, and the fitness function is parametrically defined in a way that permits to give a different weight to the cost, the time, the energy and the quality of service depending on what interests more.

The rest of the paper is structured as follows. Section 2 presents the overall software architecture. In Section 3, the GP system and the taxonomy used to build the decision model for the task of the offloading are described. In Section 4, we survey existing works. Finally, Section 5 concludes the work.

2 The Framework for Simulating and Generating Decision Models

The main idea behind our system is using Genetic Programming to evolve models, in form of Decision Trees, which will decide if it is convenient to perform the offloading of a mobile application on the basis of the parameters and the properties typical of the application, of the user and of the environment. The overall software architecture of the system is described in figure 1.

On the top of the architecture, the modules containing the data, which will be used by the other components of the system, are illustrated. The different modules will constitute a set of data for each of the four different categories considered. We remind to the section 3.1 for a detailed description of these categories. It is out of the scope of this paper to describe the techniques used to estimate these components; we just would like to mention that in the case of the parameters describing

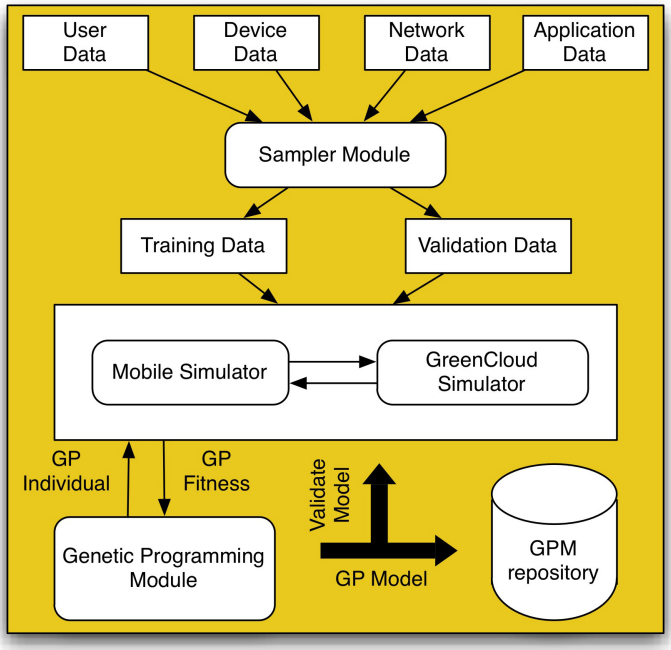


Fig. 1. The overall software architecture of the system

the mobile devices, they can be found in literature for some models and can be estimated using appropriate applications, such as PowerTutor, available in the Google Play store for the android-based mobile devices. The process to estimate the energy consumption for various types of applications and for multiple classes of devices is analogous to that presented in [11].

Afterwards, the sampler module will generate the training and the validation dataset, simply combing randomly the data estimated by the above-mentioned models. These two datasets will be used respectively to generate and validate the decision models.

Analyzing the rest of the software architecture, we find the two main modules, used respectively for the simulation and for the inference of the mobile offloading module. The inference part of the designed system consists of a Genetic Programming module, developing a population of models, apt to decide the possible offloading of a mobile application. The chosen GP system is CAGE (CellulAr GEnetic programming), a scalable cellular implementation of GP [4]. One of the advantages of the chosen GP–based module is that it can run on parallel/distributed architectures, permitting time-saving in the most expensive phase of the training process, described in the following. Indeed, each single model of the GP population represents a decision tree able to decide a strategy for the offloading and must be evaluated using the simulation module.

The simulation module consists of the GreenCloud simulator (simulating the cloud part of the offloading process) together with a mobile simulator built ad-hoc, modeling the mobile device's behavior. In practice, each model generated by the GP module is passed to the simulator module, which provides to perform the fitness evaluation directly on the basis of the results obtained simulating the model using the training dataset.

At the end of the process, the best model (or the best models) will constitute the rules adopted by the offloading engine, which will decide whether an application must be offloaded, considering the defined conditions (user requirements, bandwidth, characteristic of the mobile device and so on). All these models must be validated using the simulation engine with the validation dataset; if the result of this evaluation overcomes a predefined threshold, the model is added to a GP model repository for future use.

3 Designing an Efficient GP System for Automatic Mobile Offloading

In this section, the GP system and its design are described.

The GP system chosen for generating the rules used to perform the offloading is the parallel Genetic Programming tool CAGE [4]. The motivation behind this choice is that this system can run on parallel/distributed architectures, permitting time-saving in the most expensive phase of the training process; in addition, GP supplies the characteristic of adaptivity and the possibility of working with a few knowledge of the domain, really useful to this particular aim. As usual, in order to use GP for a particular domain, it is sufficient to choice an appropriate terminal and function set and to define a fitness function. We chose a typical approach to GP for generating decision trees, choosing as terminals simply the two answers, yes or no to the question "Is the process offloadable?".

In the following subsection, the taxonomy of the main parameters/properties of the offloading process is described. This taxonomy will be used to define the functions of the GP module, while the terminals would be constituted by the final decision on the offloading process. In the subsection 3.2, the way in which the fitness is evaluated is described.

3.1 A Taxonomy of the Main Properties for the Offloading Process

This section is devoted to the definition of a taxonomy of the parameters and of the properties, which the GP module will use to build the model that will decide the offloading strategy. Our taxonomy divides the parameters in four different categories: Application (i.e. the parameters associated to the application itself), User (the parameters desired and imposed by the user's needs), Network (i.e. parameters concerning the type and the state of the network), and Device (i.e. the parameters depending only from the hardware/software features of the devices). In practice, the decision model, built by the GP module of our architecture, will perform the decision of offloading or not on the basis of the different parameters associated to

these categories. It is worth to notice that many parameters could be more detailed and complex and other could be added; however, our taxonomy does not pretend to be exhaustive, but we tried to simplify the model and did not consider particular aspects, which should not influence the offloading process.

On the following, we will give a short description of the parameters chosen for each defined category. Where not differently specified, the values of the parameters have been discretized in ranges, using discrete values such as *low*, *moderate*, *high*, etc.

Application

The parameters associated to this category consider features, which are typical of the application to be executed on the mobile devices, trying to characterize all the aspects useful to take a decision on the offloading process.

Average Execution Time. The average execution time of the application measured for different typical dimensions of the input.

Memory. The size of the code and of the data of the application.

Local Computation/communication ratio. A value expressing if the application devotes most of its execution time to the execution of the local computation or to use the network (0 indicates an application performing only local computation, 1 an application continuously interacting with the network).

Probability of Interruption. A mobile application could be interrupted for different reasons: system crash, user interruption, no availability of a necessary resource (network, gps, etc.). This parameter represents the probability the application is interrupted before the end of the process.

User Requirements

This class of parameters considers the needs and the behavior of the user of the mobile device, modeling the different categories of users with their different styles.

Mobility. Every user has different behaviors and habits in terms of mobility. Some users spend most of the time in the same place (typically at house and at work), while other moves frequently in different places and this should be considered in the process of offloading for the continuous changes in the network used, the need for a better duration of battery and so on. This parameter models the probability of mobility of the user.

Urgency. This parameter models the urgency or the priority with which the user wants to obtain the partial/final results of the application. If the user is too impatient to get results could prefer a greater consumption of the battery rather than a larger time of waiting.

Privacy sensitivity. People are understandably sensitive about how the application captures their data and how the data are used. However, different users can

present different sensibility, from the paranoiac to the too confident user. In our context, choosing solutions privacy-preserving can degrade the performance of the offloading process, mainly for the difficulty of moving the data or for the additional cost of adding protection.

Cost. The cost a user can/wants to pay is a fundamental parameter in order to perform the offloading. In fact, the cost of the cloud platform is usually correlated with the time and the resources used.

Network

The network plays a fundamental role in the process of offloading, as it determines the velocity of the process and limits the possibility of exchanging data in real time. In fact, application needing to exchange a stream of data cannot be offloaded whether the networks do not permit a fast transfer of the data.

QoS of the network. We consider as Quality of Service of the network a parameter estimating the reliability and stability of the network.

Bandwidth. The bandwidth of the network is an important parameter to establish if the offloading of large applications/data can be quickly executed.

Latency. The latency of the network is useful when small quantities of data must be exchanged during the offloading process.

Type of network. This parameter models the type of network available, i.e. Wifi, 3G, 2G, etc. The type of the network is crucial for saving energy, as for instance, a device transmitting data using wifi consumes less battery than using 3G.

Mobile Device

The last class of parameters we consider is pertinent to the mobile device. Analyzing the state and the characteristics of the mobile device is essential in order to drive the process of offloading. On the following we list the most important parameters to be considered.

Battery Level. The battery charge level is of fundamental importance, as, when it is low, the process of offloading can be encouraged.

CPU Load and Memory availability. The load of the cpu and the available memory are two important parameters, which can influence the choice of the offloading rather than the execution on local resources.

Connectivity. This parameter represents the strength of the network signal detected by the mobile device (obviously it depends both from the device and from the quality of the network).

Example of Rules

After the description of the different parameters of the four categories defined above, we give two examples of rules, which could be extracted from our system

Table 1. Example of rules generated for a chess game and for a password manager application

Password Manager	**if** memory is high **and** privacy is high **and** bandwidth is low **then** execute locally
Chess Game	**if** privacy is low **and** bandwidth is high **and** cost is moderate **then** perform offloading

concerning two typical applications, i.e. a password manager and a chess game, shown in table 1. Note that, really, even if our framework generates decision trees, we choose to show them in terms of $if \ldots then$ rules, for the sake of clearness.

3.2 Fitness Evaluation

This subsection defines the fitness function used in the GP inference engine. We modeled it with a simple equation, described in the following.

First of all, we define three normalized functions, representing respectively the energy saved, the time saved and the cost saved during the process of offloading (really the latter is a negative value, as it is a cost not a saving): S_{energy}, S_{time} and S_{cost}.

$S_{energy} = \frac{E_{local} - E_{offload}}{max(E_{offload}, E_{local})}$, i.e. the ratio between the energy saved executing the process on remote servers and the energy necessary to perform the offloading. The energy is computed in accordance with the analysis defined in [7].

$S_{time} = \frac{T_{local} - T_{offload}}{max(T_{offload}, T_{local})}$, i.e. the ratio between the time saved executing the process on remote servers and the time necessary to perform the offloading.

Differently, the cost function is computed as $S_{cost} = -\frac{C_{offload}}{C_{sup}}$, i.e. the ratio between the cost due to the remote execution and a parameter C_{sup} defining a threshold of cost (if the cost overcomes C_{sup} , S_{cost} becomes -1).

Finally, the fitness is computed as the weighted sum of the three equations described above, using three positive parameters (a, b , c), modeling the importance we want to give respectively to the energy saving, to the time saving and to the cost saving.

Considering an element T_i (representing an application running on a determined device, with a required QoS, etc.) of the training set T composed by n tuples, the fitness of this element is computed as

$f(T_i) = a * S_{energy} + b * S_{time} + c * S_{cost}$ and consequently the total fitness is given by $f_{tot} = \sum_{i=1}^{i=n} f(T_i) - d * QoS$

where the term QoS represent the ratio between the number of cases (tuples) of the datasets, which does not respect the QoS constraints, and the total number of tuples. In practice, a penalty is added to the fitness function for each case in which the QoS is not guaranteed. A fourth parameter d is used to give a particular weight to the QoS.

4 Related Works

Analyzing the works in literature concerning the offloading of mobile applications, the problematic of finding an automatic methodology to perform offloading is not much explored.

A paper introducing general thematic on the process of offloading is written by Kumar and Lee [7]. The authors analyze the main problems derived from offloading mobile applications on the Cloud such as privacy, costs, energy consumption and show which applications can benefit from this approach. They introduce a simple model for deciding whether it is convenient to perform the offloading and they try to apply the technique only to computationally expensive functions while computing locally other tasks.

Other papers are devoted to the utility of performing offloading, basing on some criteria, i.e. energy consumption, costs, network usage, etc. For instance, in [10] the decision of performing the offloading is based on the difference between the energy used if the task was executed locally on the mobile device or remotely on the cloud servers. The power consumption of the local execution is estimated by counting the cpu cycles while, as for the remote execution, it is calculated only considering the network usage (data transfer). Our model is more sophisticated, as it considers also the hardware components that are used during computation and the problematic concerning the transfer of the data (i.e. cpu, wifi, 3g, display, system, etc.).

In [6] a two-step method is used. First, a database of application power usage is built through standard profiling technics. Then, the authors exploit the property stated in the paper that, for a limited class of applications (i.e. applications in which the cost depends only on the scale of the input), the algorithmic complexity combined with the profiling can be used to predict the energy cost of the execution of the application itself. Unfortunately, real-world applications can be hardly modeled considering only their input.

Many papers are devoted to techniques and strategies to alleviate the process of offloading analyzing the code of the application or optimizing some energy-consuming processes, i.e. the acquisition of the GPS signal. For instance, Saarinen et al. [14] analyses the application source code and identifies method presenting hardware and/or software constraints, which do not permit the offloading. In addition, they also consider traffic patterns and power saving modes. However, the work does not consider network conditions and user requirements. Note that these approaches are orthogonal to our work and can be adopted in order to optimize some phases of the offloading process.

Spectra [3] is a remote execution system that monitors application resource usage and the availability of the resources on the device and dynamically choose how and where to execute application tasks. The framework provides APIs to developers to build application suitable with the defined architecture. Xray [13] is an automatic system, which profiles an application and decides what and when an offloading computation is useful. The Xray profiling stage observes application and system events (Gui, sensor, GPS, memory, CPU) and identifies remotable methods. If a method does not use local resources then it is remotable. Differently

form these two systems, our framework is not method-based, but considers the entire application and the decision to perform the offloading is based not only on the application characteristics (size of data, privacy concern) but also on the system status (battery, 3g or wifi connection) and on some constraints requested by the user.

Gu et al. [5] extend an offloading system with an offloading inference engine (OLIE). OLIE solves two problems: first, it decides when to trigger the offloading action; second, it selects a partitioning policy that decides which objects should be offloaded and which pulled back during an offloading action. The main focus of OLIE is to overcome memory limitations of a mobile device. To make a decision OLIE consider available memory and network conditions (i.e. bandwidth, delay). To achieve a more powerful triggering system, OLIE uses a Fuzzy Control model with rules specified by the system and by the application developers. Similarly to our approach, this paper is based to an evolutionary algorithm to automate the decision of offloading a task in order to save energy. However, it only considers the hardware/software characteristics of the application and of the device and not the user's requirements and the QoS.

5 Conclusions and Future work

This work presents an automatic approach to generate models for taking decisions on the process of offloading of mobile applications on the basis of the user requirements, of the conditions of the network, of the hardware/software features of the mobile device and of the characteristics of the application. The system constitutes a general framework for testing offload algorithms and includes a mobile simulator and an inference engine. The latter is a genetic programming module, in which the functions are the parameters of a ad-hoc designed taxonomy, which includes all that can influence the process of offloading and that must be considered during the decision process. The fitness function is parametrically defined in a way that permits to give a different weight to the cost, the time, the energy and the quality of service depending on what interests more.

For future works, the framework will be tested with real datasets in order to verify whether the models work in real environments. Finally, the taxonomy will be refined on the basis of the results obtained from the experiments.

Acknowledgments, This research work has been partially funded by the MIUR project FRAME, PON01-02477.

References

1. Abowd, G.D., Dey, A.K.: Towards a Better Understanding of Context and Context-Awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999)
2. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25(6), 599–616 (2009)

3. Flinn, J., Park, S., Satyanarayanan, M.: Balancing performance, energy, and quality in pervasive computing. In: ICDCS, pp. 217–226 (2002)
4. Folino, G., Pizzuti, C., Spezzano, G.: A scalable cellular implementation of parallel genetic programming. IEEE Transaction on Evolutionary Computation 7(1), 37–53 (2003)
5. Gu, X., Nahrstedt, K., Messer, A., Greenberg, I., Milojicic, D.S.: Adaptive offloading inference for delivering applications in pervasive computing environments. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), Fort Worth, Texas, USA, March 23-26, pp. 107–114 (2003)
6. Gurun, S., Krintz, C.: Addressing the energy crisis in mobile computing with developing power aware software. In: UCSB Technical Report. UCSB Computer Science Department (2003)
7. Kumar, K., Lu, Y.-H.: Cloud computing for mobile users: Can offloading computation save energy? IEEE Computer 43(4), 51–56 (2010)
8. Lee, K., Rhee, I., Lee, J., Chong, S., Yi, Y.: Mobile data offloading: how much can wifi deliver? In: CoNEXT 2010, Philadelphia, PA, USA, November 30 - December 03, p. 26. ACM (2010)
9. Liu, J., Kumar, K., Lu, Y.-H.: Tradeoff between energy savings and privacy protection in computation offloading. In: Proceedings of the 2010 International Symposium on Low Power Electronics and Design, Austin, Texas, USA, August 18- 20, pp. 213–218. ACM (2010)
10. Miettinen, A.P., Nurminen, J.K.: Energy efficiency of mobile clients in cloud computing. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2010, USENIX Association, Berkeley (2010)
11. Namboodiri, V., Ghose, T.: To cloud or not to cloud: A mobile device perspective on energy consumption of applications. In: WOWMOM, pp. 1–9 (2012)
12. Ortiz, A., Ortega, J., Díaz, A.F., Prieto, A.: Modeling network behaviour by full-system simulation. JSW 2(2), 11–18 (2007)
13. Pathak, A., Hu, Y.C., Zhang, M., Bahl, P., Wang, Y.-M.: Enabling automatic offloading of resource-intensive smartphone applications. Technical report, Purdue University (2011)
14. Saarinen, A., Siekkinen, M., Xiao, Y., Nurminen, J.K., Kemppainen, M., Hui, P.: Offloadable apps using smartdiet: Towards an analysis toolkit for mobile application developers. CoRR, abs/1111.3806 (2011)
15. Wolski, R., Gurun, S., Krintz, C., Nurmi, D.: Using bandwidth data to make computation offloading decisions. In: 22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida, USA, April 14-18, pp. 1–8. IEEE (2008)

Solving the Location Areas Scheme in Realistic Networks by Using a Multi-objective Algorithm

Víctor Berrocal-Plaza, Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez, and Juan A. Gómez-Pulido

Dept. Technologies of Computers & Communications, University of Extremadura
Escuela Politécnica, Campus Universitario S/N, 10003, Cáceres, Spain
{vicberpla,mavega,sanperez,jangomez}@unex.es

Abstract. The optimization of the management tasks in current mobile networks is an interesting research field due to the exponential increase in the number of mobile subscribers. In this paper, we study two of the most important management tasks of the Public Land Mobile Networks: the location update and the paging, since these two procedures are used by the mobile network to locate and track the Mobile Stations. There are several strategies to manage the location update and the paging, but we focus on the Location Areas scheme with a two-cycle sequential paging, a strategy widely applied in current mobile networks. This scheme can be formulated as a multi-objective optimization problem with two objective functions: minimize the number of location updates and minimize the number of paging messages. In previous works, this multi-objective problem was solved with single-objective optimization algorithms by means of the linear aggregation of the objective functions. In order to avoid the drawbacks related to the linear aggregation, we propose an adaptation of the Non-dominated Sorting Genetic Algorithm II to solve the Location Areas Planning Problem. Furthermore, with the aim of studying a realistic mobile network, we apply our algorithm to a scenario located in the San Francisco Bay (USA). Results show that our algorithm outperforms the algorithms proposed by other authors, as well as the advantages of a multi-objective approach.

Keywords: Location Areas Planning Problem, Non-dominated Sorting Genetic Algorithm II, Mobile Location Management, Multi-objective Optimization, Stanford University Mobile Activity Traces.

1 Introduction

The Public Land Mobile Networks (PLMNs) are the networks that provide mobile communications to the public. These networks can be represented by a hierarchical structure with three levels: Mobile Subnet, Radio Network, and Core Network [1]. The first level (Mobile Subnet) is constituted by the subscriber terminals (or Mobile Stations, MS). The second level (Radio Network or Access Subnet) divides the coverage area in several smallest regions, known as cells, among which the radio-electric resources are distributed and reused. This level is constituted by the Base

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 72–81, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Stations (BSs) and the Base Station Controllers (BSCs). A BS is the network entity that provides mobile access to the MSs, and a BSC is the network entity that performs control tasks associated with several BS. And the third level (Core Network) comprises all systems and registries that perform the management, control and operation tasks, e.g. the Mobile Switching Centers (MSCs), the Home Location Registries (HLRs), and the Visitor Location Registries (VLRs) are the entities responsible for managing the subscriber location update (LU) and paging (PA). These management tasks (LU and PA) are used by the PLMNs to know the exact location of the subscriber (in terms of cells) in order to automatically route an incoming call to him, so they are considered as two of the most important tasks in the PLMNs [2].

By using the LU procedure, the MS notifies the mobile network that its location should be updated. There are several strategies to manage the LU, mainly classified into two groups: static LU and dynamic LU [3-5]. In static strategies, all subscribers perceive the same logical topology of the mobile network, which is calculated by the Location Management system (the system that controls the location update and the paging). However, in dynamic strategies, each MS might perceive a different logical topology and it decides to perform a location update according to its calling and mobility patterns. With these last strategies, the signaling load associated with the LU and PA procedures might be reduced considerably, but they require higher network capabilities than the static strategies, since the mobile network has to store a logical configuration per subscriber. That is why the static strategies are more popular than the dynamic ones. Always Update, Never Update, and Location Areas are examples of static LU strategies [3]. In the Always Update strategy, the MS updates its location whenever it moves to a new cell, and hence, the paging is only performed in the current cell of the callee user. In contrast to the Always Update strategy, there is not location update in the Never Update strategy, and thus, all network cells have to be polled in order to know the location of the callee user. The Location Areas (LA) strategy is halfway between the Always Update and the Never Update strategies. By means of the Location Areas scheme, the network cells are grouped into logical areas such that the MS only updates its location when it moves to a new logical area, and the paging is only performed in the cells within the logical area of the callee user.

On the other hand, the mobile network uses the PA procedure to know the exact cell associated with the callee user [4]. Commonly, this procedure has to be performed before the timer expires (known as maximum paging delay). Two-cycle Sequential Paging, Blanket Polling, and Shortest Distance are examples of paging procedures with delay constraint. In the Two-cycle Sequential Paging, the cells that have to be polled are grouped into two paging areas that are polled sequentially. First, the system checks the last cell known for the callee user, and then, if it is necessary, the rest of cells of that location area. This is one of the PA procedures more used in mobile networks. Fig. 1 shows an example of the LU and PA procedures.

In the last decade, due to the exponential increase in the number of mobile subscribers, several authors have focused their researches on applying new optimization techniques to the Location Management tasks, and concretely in the Location Areas scheme. P. R. L. Gondim is one of the first authors arguing that the Location Areas Planning Problem is an NP-hard combinatorial optimization problem

(due to the huge size of the objective space), and he defined a Genetic Algorithm (GA) for finding quasi-optimal configurations of Location Areas [6]. P. Demestichas et al. proposed three algorithms (GA, Simulated Annealing (SA), and Taboo Search (TS)) to research the Location Areas scheme in different environments [7]. J. Taheri and A. Zomaya implemented several algorithms to solve the Location Areas Planning Problem: Hopfield Neural Networks (HNNs), GA, SA, and combinations of GA with HNN (GA-HNN) [8-11]. R. Subrata and A. Zomaya proposed a dynamic LU strategy based on the Location Areas scheme [12]. In their work, R. Subrata and A. Zomaya use the network provided by the Stanford University (SUMATRA [13]: Stanford University Mobile Activity TRAces, a trace generator that is well-validated against real world data measured in the geographical area of the San Francisco Bay, USA). Finally, S. M. Almeida-Luz et al. developed the Differential Evolution (DE) algorithm [14-15] and the Scatter Search (SS) algorithm [16-17] to also solve the SUMATRA network [13].

In all of these algorithms, the objective functions of the Location Areas Planning Problem are linearly combined into a single objective function. This technique reduces the complexity of the optimization algorithm but has got associated several drawbacks, as can be seen in Section 3. That is why we propose an adaptation of the Non-dominated Sorting Genetic Algorithm II (NSGA-II, a Multi-Objective Evolutionary Algorithm) to solve the Location Areas Planning Problem. With a multi-objective optimization algorithm, we can obtain a set of solutions among which the network operator could select the one that best fits the network real state, i.e. when the signaling load generated by other management systems is considered.

The paper is organized as follows: Section 2 defines the Location Areas Planning Problem and presents the SUMATRA network that we have studied. Section 3 shows the main features of a multi-objective optimization algorithm and our adaptation of NSGA-II to solve the Location Areas Planning Problem. Results, comparisons with other authors, and an analysis of the obtained solutions are presented in Section 4. Finally, our conclusions and future work are discussed in Section 5.

2 Location Areas Scheme

The Location Areas scheme is being widely used in current mobile networks. By means of this strategy, the network cells are grouped into logical areas (or Location Areas, LAs) with the aim of reducing the number of signaling messages associated with the subscriber location update, since the MS is free to move inside a given LA without updating its location. Furthermore, the paging procedure is only performed in the cells within the current LA of the callee user. It should be noted that the main challenge of the Location Areas scheme is to find the LA configuration that minimizes simultaneously the number of location update and the number of paging messages.

To reduce the number of location updates, the size of the LAs should be increased, leading to an increment in the number of paging messages because more cells have to be paged. And vice versa, to reduce the number of paging messages, the size of the

LAs should be reduced, leading to an increment in the number of location updates. Therefore, the Location Areas scheme defines a Multi-objective Optimization Problem (MOP) that can be formulated as:

$$f_1 = \min\left\{ LU_{cost} = \sum_{t=T_{ini}}^{T_{fin}} \sum_{i=1}^{N_{user}} \gamma_{t,i} \right\}, \tag{1}$$

$$f_2 = \min\left\{ PA_{cost} = \sum_{t=T_{ini}}^{T_{fin}} \sum_{i=1}^{N_{user}} \rho_{t,i} \left(\alpha_{t,i} + \left(1 - \alpha_{t,i}\right) \times NA[LA_t[i]] \right) \right\}, \tag{2}$$

subject to

$$\sum_{i=1}^{N_{Cell}} \sum_{k=1}^{N_{Area}} \mu_{i,k} = 1, \tag{3}$$

where the involved variables are:

- $\gamma_{t,i}$: A binary variable that is equal to 1 when the MS *i* moves to a new Location Area in the time *t*; otherwise $\gamma_{t,i}$ is equal to 0.
- $\rho_{t,i}$: A binary variable that is equal to 1 when the MS *i* has an incoming call in the time *t*; otherwise $\rho_{t,i}$ is equal to 0.
- $\alpha_{t,i}$: A binary variable that is equal to 1 when, in the time *t*, the MS *i* is located in its last updated cell; otherwise $\alpha_{t,i}$ is equal to 0.
- NA: Vector that stores the size (in terms of number of cells) of each Location Area.
- LA_t: Vector that stores the Location Area associated with each user in the time *t*.
- $\mu_{i,k}$: A binary variable that is equal to 1 when the cell *i* belongs to the Location Area *k*; otherwise $\mu_{i,k}$ is equal to 0.
- N_{user}: Number of mobile users.
- N_{Cell}: Number of network cells.
- N_{Area}: Number of Location Areas.
- $[T_{ini}, T_{fin}]$: Time interval during which the LU and PA costs are calculated.

Equation (1) defines the first objective function of the Location Areas Planning Problem: minimize the number of location updates. Equation (2) shows the second objective function: minimize the number of paging messages required to locate a callee user by using the Two-cycle Sequential Paging. In this work, we use the paging procedure proposed in [12, 15, 17]: firstly, the callee MS is searched in the last updated cell and, if the MS is not found in this cell, the other cells of the Location Area are simultaneously polled in order to know the exact cell in which the MS is located. Constraint (3) establishes that a cell cannot be associated with several Location Areas, and has to be associated always with a Location Area. Therefore, the maximum Location Area size is limited to NCell (i.e. the Never Update strategy, when all cells belong to the same Location Area), and the maximum number of Location Areas is also limited to NCell (i.e. the Always Update strategy, when each cell belongs to a different Location Area).

In previous works, the Location Areas Planning Problem was solved by means of Single-objective Optimization Algorithms (SOA). For it, the linear aggregation of the objective functions was used to adapt this multi-objective optimization problem to a single-objective optimization problem. Equation (4) shows the objective function proposed in [8-12, 14-17], where β is a constant defined to consider that the cost associated with a location update is higher than the cost of performing the paging procedure, since the location update involves more network entities than the paging. Commonly, this coefficient is configured equal to 10, β = 10.

$$f^{SOA} = \min\left\{\text{Cost}_{tot}(\beta) = \beta \times \text{LU}_{cost} + \text{PA}_{cost}\right\}, \quad (4)$$

In this paper, with the aim of avoiding the drawbacks associated with the linear aggregation (see Section 3), we propose a multi-objective optimization algorithm to solve the Location Areas Planning Problem. With this strategy, the network operator could select the solution that best adjusts to the network real state.

2.1 Stanford University Mobile Activity Traces

The Stanford University Mobile Activity TRAces (SUMATRA) is a set of mobile activity traces that are available to the public via web [13]. In this work, we have studied the BALI-2 network, which provides a real-time data of the users' mobile activity measured in the San Francisco Bay (USA) during 24 hours. BALI-2 defines a mobile network with 90 cells and 66,550 subscribers. The main appeal of this network is that it will allow us to study the behavior of the Location Areas Scheme in a realistic mobile network, since its trace is well validated against real world data. Fig. 2 shows an approximation of the BSs planning and its associated graph. Note that each circle represents a BS and the edges represent the neighborhood among cells (i.e. two cells are neighboring if they are connected by an edge).

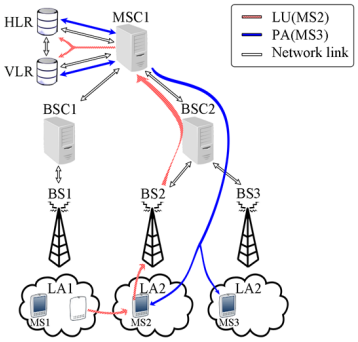


Fig. 1. Example of location update and paging in the Location Areas scheme

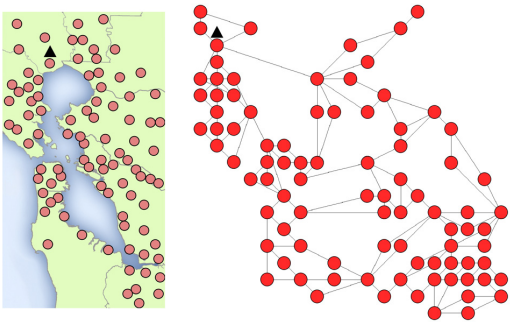


Fig. 2. BALI-2 mobile network: BS planning and associated graph

3 Multi-objective Optimization Paradigm

A Multi-objective Optimization Problem (MOP) is the optimization problem in which two or more conflicting objective functions have to be optimized under certain constraints [18], e.g. the Location Areas Planning Problem. Due to the fact that there are two or more conflicting objectives, the main task of a multi-objective optimization algorithm consists in finding a set of solutions, each one related to a trade-off among objectives. Commonly, these similar-quality solutions are called non-dominated solutions and they are grouped in the known as Pareto front. For definition, the solution $\mathbf{x}^1$ is said to dominate the solution $\mathbf{x}^2$ (denoted by $\mathbf{x}^1 \prec \mathbf{x}^2$) when $\mathbf{x}^1$ outperforms $\mathbf{x}^2$ in one or more objectives, and $\mathbf{x}^1$equals $\mathbf{x}^2$ in the rest of objectives.

Traditionally, the linear aggregation of the objective functions was used to solve a MOP by means of single-objective optimization algorithms, e.g. see Equation (4). This strategy allows the use of less complex algorithms but has got associated several drawbacks. Firstly, an accurate knowledge of the problem is required in order to properly configure the values of the weight coefficients, which could be real values. Secondly, the proper value of the weight coefficients might vary in the time (e.g. in the Location Areas Planning Problem, different states of the signaling network could require different values of β). And thirdly, a single-objective optimization algorithm must perform an independent run for each combination of the weight coefficients.

In this paper, we have implemented an adaptation of the Non-dominated Sorting Genetic Algorithm II to solve the Location Areas Planning Problem. A detailed explanation of the adaptation of this Multi-Objective Evolutionary Algorithm (MOEA) is discussed in Subsection 3.1.

3.1 Non-dominated Sorting Genetic Algorithm II

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a population-based metaheuristic algorithm in which the EVolutionary OPerators (EVOPs: recombination of parents or crossover, mutation, and natural selection of the fittest individuals) of biological systems are used to iteratively improve a set of solutions. This algorithm was proposed by K. Deb et al. in [19] with the goal of reducing the computational complexity of its predecessor NSGA [20]. The pseudo-code of our adaptation of NSGA-II is presented in Algorithm 1. In this pseudo-code: Npop is the population size, P_C is the crossover probability, P_M is the mutation probability, and the stop condition is the Maximum Number of Cycles (MNC) executed.

Firstly, the population should be initialized and evaluated, i.e. a solution must be calculated and evaluated for each individual. In this work, every individual (its genome) is represented by a vector that stores the Location Area associated with each network cell. And secondly, the EVOPs are applied iteratively until the stop condition is not satisfied. The crossover operation is used to generate the offspring population, where each offspring has genetic information of two parents. We have used an elitist crossover, in which the number of crossover points and their positions are randomly determined between [1, 4] and [0, NCell-1], respectively. The mutation operation is applied to change the genetic information of the offspring. We have defined two

mutation operations: Gene Mutation (GM) and Merge-LA Mutation (MLAM). The Gene Mutation consists in changing the Location Area of a boundary cell by its neighboring Location Area of lower size (in terms of number of cells). And the Merge-LA Mutation consists in merging the smallest Location Area with its neighboring Location Area with fewer cells. These two mutation operations cannot be applied simultaneously over the same individual, and they are configured such that $2P_{GM} = 2P_{MLAM} = P_M$, where P_{GM} is the Gene Mutation probability and P_{MLAM} is the Merge-LA Mutation probability. Finally, the natural selection is performed with the goal of selecting the Npop fittest individuals, which will be the parent population in the next cycle. K. Deb et al. [19] provide a methodology to select the best solutions (or individuals) in the multi-objective context, i.e. they provide a multi-objective fitness function. This methodology consists of two functions: the Non-dominated Sorting and the Crowding Distance. The Non-dominated Sorting is a function that applies the non-dominance concept to arrange a set of solutions in groups (or fronts). And the Crowding Distance measures the density of solutions surrounding a particular point of the objective space. For detailed information about NSGA-II, please see [19].

In order to perform a fair comparison with other works, we use the same stop condition (MNC=1000) and the same population size (Npop=300) [15]. The other parameters of NSGA-II have been configured by means of a parametric study, in which we have performed 30 independent runs per experiment. The parameter configuration that provides the higher Hypervolume is: $P_C = 0.9$, $P_M = 0.2$. The Hypervolume is one of the most popular multi-objective indicators, and it is used to know the quality of a multi-objective optimization algorithm. This indicator measures the area of the objective space covered by the Pareto front (if we assume two objectives).

Algorithm 1. Pseudo-code of our adaptation of NSGA-II

```
1:   % Initialization of the population
2:   Indv ← Initialization(Npop);
3:   % Evaluation of the population
4:   Indv ← ObjectiveFunctionsEvaluation(Indv);
5:   Indv ← MOFitnessFunctionEv(Indv);
6:   % Main loop
7:   while (stop condition ≠ TRUE){
8:   % Crossover or recombination of parents
9:   Offsp ← Crossover(Indv,P_C);
10:  % Mutation of the offspring
11:  Offsp ← Mutation(Offsp,P_M);
12:  % Evaluation of the offspring
13:  Offsp ← ObjectiveFunctionsEvaluation(Offsp);
14:  % Evaluation of all population
15:  [Indv,Offsp] ← MOFitnessFunctionEv(Indv,Offsp);
16:  % Selection of the fittest individuals
17   Indv ← NaturalSelection(Indv,Offsp);
18:  }
```

4 Experimental Results

With the purpose of checking the behavior of our algorithm in a realistic mobile network, we have studied one of the networks developed by the Stanford University: BALI-2, the network that provides real-time information of the mobile activity. Furthermore, in order to verify the quality of our solutions, we have compared our results with those obtained by other authors. Due to the fact that there is no other work that addresses the Location Areas Planning Problem in a multi-objective manner, we must compare with single-objective optimization algorithms [15, 17]. For it, we have to find in our Pareto front the solution that best fits the Equation (4) with β equal to 10, since it is the objective function used by these single-objective optimization algorithms. In addition, we have calculated statistical data of the Hypervolume (median, HV_{median}, and interquartile range, HV_{iqr}) for 30 independent runs. These statistical data are: $HV_{median}(\%) = 93.75\%$, $HV_{iqr} = 7.8700e^{-4}$. It should be noted the low value of the HV_{iqr}, this denotes that our algorithm is very stable. Fig. 4 shows the Pareto front related to the median HV, the reference points that we have used to calculate the HV (these reference points have been calculated by means of the extreme solutions: Always Update strategy and Never Update strategy, see Section 2), and the solution of our Pareto front that best fits the Equation (4) with β equal to 10. In this figure, we can observe the high spread of our Pareto front, which includes the two extreme solutions. Fig. 3-(a) presents a comparison among the Always Update strategy, the Never Update strategy, and our proposal. This figure shows us that our algorithm clearly outperforms the two classic strategies. Finally, Fig. 3-(b) shows a comparison between our algorithm (NSGA-II) and the algorithms proposed in [15] and [17]: Differential Evolution (DE) and Scatter Search (SS). In this figure we can see that our algorithm outperforms these two single-objective optimization algorithms in the hours with higher mobile activity (8:00h-20:00h). A summary of these two last figures is presented in Table 1, in which we can observe that our algorithm also obtains better results than one of the dynamic strategies proposed in [12]: Distance-Based Location Area (DBLA). Therefore, we can conclude that our algorithm is competitive, since it achieves better results than other algorithms and it provides a set of solutions among which the network operator could select the one that best adjusts to the network real state.

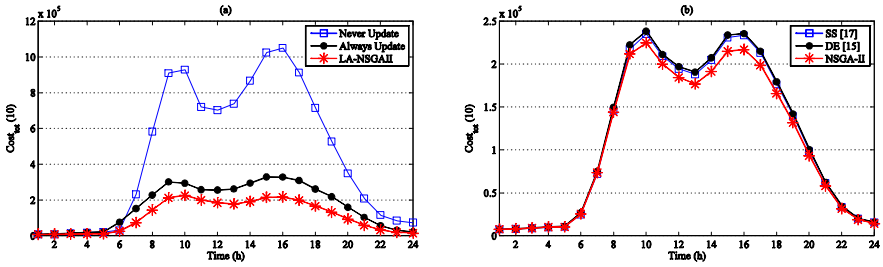


Fig. 3. BALI-2: (a) Comparison among strategies. (b) Comparison among algorithms.

Table 1. Comparison among strategies

Algorithm	$\sum_{t=0h}^{24h} Cost_{tot}(10)$
LA-NSGA-II	2,619,519
LA-SS[17]	2,756,836
LA-DE[15]	2,799,289
Always Update	4,010,317
Never Update	10,836,953
DBLA[12]	2,695,282

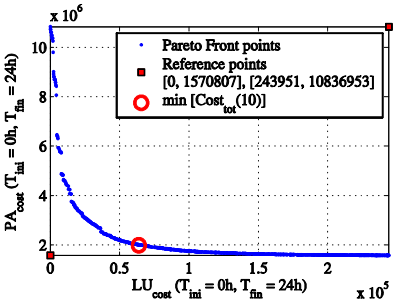


Fig. 4. Median Pareto front of NSGA-II

5 Conclusions and Future Work

The main contribution of this work is that we have adapted the Non-dominated Sorting Genetic Algorithm II (NSGA-II, a Multi-Objective Evolutionary Algorithm) to solve the Location Areas Planning Problem in a realistic mobile network: BALI-2 [13]. Due to the fact that there is no other work that addresses this problem with a multi-objective optimization algorithm, we must compare with single-objective optimization algorithms. Results show that our algorithm is promising, since it achieves better solutions than the algorithms proposed by other authors, and it gives more vision to the network operator. In a future work, it would be interesting to develop other multi-objective optimization algorithms and compare them with NSGA-II. Furthermore, it would be a good challenge to study the Location Areas Planning Problem when the network architecture is considered.

Acknowledgement. The work of Víctor Berrocal-Plaza has been developed under the Grant FPU-AP2010-5841 from the Spanish Government. This work was partially funded by the Spanish Ministry of Economy and Competitiveness and the ERDF (European Regional Development Fund), under the contract TIN2012-30685 (BIO project).

References

1. Agrawal, D.P., Zeng, Q.A.: Introduction to Wireless and Mobile Systems, 3rd edn. Cengage Learning, Stamdford (2010)
2. Garg, V.: Wireless Communications and Networking, 1st edn. Elsevier, San Francisco (2007)
3. Tabbane, S.: Location management methods for third-generation mobile systems. IEEE Commun. Mag. 35(8), 72–84 (1997)
4. Wong, V.W.-S., Leung, V.C.M.: Location management for next generation personal communications networks. IEEE Network 14(5), 18–24 (2000)

5. Kyamakya, K., Jobmann, K.: Location management in cellular networks: classification of the most important paradigms, realistic simulation framework, and relative performance analysis. IEEE Transactions on Vehicular Technology 54(2), 687–708 (2005)
6. Gondim, P.: Genetic algorithms and the location area partitioning problem in cellular networks. In: Proceedings of IEEE 46th Vehicular Technology Conference on Mobile Technology for the Human Race, vol. 3, pp. 1835–1838 (1996)
7. Demestichas, P., Georgantas, N., Tzifa, E., Demesticha, V., Striki, M., Kilanioti, M., Theologou, M.: Computationally efficient algorithms for location area planning in future cellular systems. Computer Communications 23(13), 1263–1280 (2000)
8. Taheri, J., Zomaya, A.: The use of a Hopfield neural network in solving the mobility management problem. In: Proceedings of The IEEE/ACS International Conference on Pervasive Services, pp. 141–150. IEEE Computer Society, Washington (2004)
9. Taheri, J., Zomaya, A.: A genetic algorithm for finding optimal location area configurations for mobility management. In: Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary, pp. 568–577. IEEE Computer Society, Washington (2005)
10. Taheri, J., Zomaya, A.: A Simulated Annealing approach for mobile location management. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, pp. 194–201. IEEE Computer Society, Washington (2005)
11. Taheri, J., Zomaya, A.: A combined genetic-neural algorithm for mobility management. Journal of Mathematical Modeling and Algorithms 6(3), 481–507 (2007)
12. Subrata, R., Zomaya, A.Y.: Dynamic Location Area Scheme for Location Management. Telecommunication Systems 22(1-4), 169–187 (2003)
13. Stanford University Mobile Activity TRAces (SUMATRA), `http://infolab.stanford.edu/sumatra` (accessed in 2012)
14. Almeida-Luz, S.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Differential evolution for solving the mobile location management. Applied Soft Computing 11(1), 410–427 (2011)
15. Almeida-Luz, S.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Applying Differential Evolution to a Realistic Location Area Problem Using SUMATRA. In: Proceedings of The Second International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 170–175. IEEE Computer Society, Washington (2008)
16. Almeida-Luz, S.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Applying Scatter Search to the Location Areas Problem. In: Corchado, E., Yin, H. (eds.) IDEAL 2009. LNCS, vol. 5788, pp. 791–798. Springer, Heidelberg (2009)
17. Almeida-Luz, S.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Solving a Realistic Location Area Problem Using SUMATRA Networks with the Scatter Search Algorithm. In: Proceedings of the Ninth International Conference on Intelligent Systems Design and Applications, pp. 689–694. IEEE Computer Society, Washington (2009)
18. Knowles, J., Thiele, L., Zitzler, E.: A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland (2006)
19. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
20. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2(3), 221–248 (1994)

The Small-World Phenomenon Applied to a Self-adaptive Resources Selection Model

María Botón-Fernández[1], Francisco Prieto Castrillo[1], and Miguel A. Vega-Rodríguez[2]

[1] Ceta-Ciemat, Dept. Science and Technology, Trujillo, Spain
{maria.boton,francisco.prieto}@ciemat.es
[2] University of Extremadura, Dept. Technologies of Computers and Communications, Cáceres, Spain
mavega@unex.es

Abstract. Small-world property is found in a wide range of natural, biological, social or transport networks. The main idea of this phenomenon is that seemingly distant nodes actually have very short path lengths due to the presence of a small number of shortcut edges running between clusters of nodes. In the present work, we apply this principle for solving the resources selection problem in grid computing environments (distributed systems composed by heterogeneous and geographically dispersed resources). The proposed model expects to find the most efficient resources for a particular grid application in a short number of steps. It also provides a self-adaptive ability for dealing with environmental changes. Finally, this selection model is tested in a real grid infrastructure. From the results obtained it is concluded that both a reduction in execution time and an increase in the successfully completed tasks rate are achieved.

Keywords: Small-world phenomenon, Optimization, Grid computing, Self-adaptability.

1 Introduction

A grid computing environment [6][5] is a distributed system which coordinates heterogeneous resources by using open standard protocols and interfaces without applying a centralized control. In this way, a grid system interconnects resources from different administration domains; by maintaining the internal security policies and the resources management software of every domain. This leads to a dynamic and changing environment. Moreover, this type of infrastructures presents a double heterogeneity: on the one hand, there are several groups of resources according to their functionalities. On the other hand, there are heterogeneous resources within a particular group because they are provided by different centres. This fact along with the grid dynamic and changing nature varies resources performance and availability, worsening applications execution performance. Nowadays, applications require real-time information of grid infrastructures for

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 82–91, 2013.
© Springer-Verlag Berlin Heidelberg 2013

dealing with the environmental changes. Consequently, it has become a challenge to efficiently perform the resources discovery, resources-application matching, resources allocation, planning and monitoring (i.e. solving resources management problems).

Recently, grid communities are addressing the *adaptation* concept for solving resources management. In this regard, manifold strategies have been presented: efficient tasks management frameworks, adaptive and autonomous grid systems, new monitoring and discovery processes, etc (see section 2). However, none of these implementations has been extended in a global way across the different grid platforms. For that reason, the present contribution proposes an *Efficient Resources Selection* (*ERS*) model which improves the infrastructure throughput without modifying it. This model is defined from the user point of view and located in the application layer, guiding applications during their deployment on grid infrastructures. Two main goals were fixed: on the one hand, a reduction of the application execution time and, on the other hand, an improvement of the successfully finished tasks rate. In this regard, we are interested in discovering the most efficient resources in the shortest possible time. That fact has motivated us to apply the *Small-world phenomenon* [10] [13] during the selection process. Concerning this algorithm, D. Watts and S. Strogatz [13] focussed on analysing a certain type of random graphs with peculiar connectivity properties. They showed that these networks could be reproduced by a random rewiring process between nodes in a lattice network. In that seminal paper, the authors proposed to define small-world networks as those holding a small value of characteristic path length, like random graphs [4], and a high clustering coefficient, like regular lattices. Regarding our proposed model, it is expected to find the most efficient resources in a small number of steps by applying this algorithm. The model is tested in a real grid infrastructure belonging to the *Spanish National Grid Initiative* (*ES-NGI*).[1] During this evaluation phase two scenarios were defined to determine if the objectives had been fulfilled (see section 4).

The rest of the paper is structured as follows. In Section 2 grid computing adaptation related work is presented. Section 3 includes the Efficient Resources Selection model description along with basic grid concepts for a better understanding of the model. Next, the model evaluation and the resulting data are discussed in Section 4. Finally, Section 5 concludes the paper and summarizes future work.

2 Related Work

As mentioned, there are some approaches in the grid community exploiting the *adaptation* concept for solving the resources management as well as for dealing with the dynamic nature of such infrastructures. In this regard, [2] describes the *AppLes* project, whose goal is to provide an adaptive ability to grid systems. A methodology for an adaptive application scheduling is proposed, which means,

[1] `http://www.es-ngi.es/`

developing and deploying distributed high-performance applications in an adaptive form. A framework for adaptive executions in Grid based on *Globus*[2] is described in [7]. It has been designed for handling grid jobs in an efficient way during execution. Here, to maintain a suitable performance level, the tasks execution adapts to the changing conditions by using new scheduling techniques.

Other work is focussed on improving the monitoring and discovery processes [8]. In this particular case, the study presents an approach for avoiding the *Information System* (*IS*)[3] overload while improving its performance. Two adaptive notification algorithms are exposed: a *sink-based algorithm* and a *utilization-based algorithm*. Both are based on *IS* availability and on data accuracy requirements. An autonomous grid system is presented in [12]. That system is dynamically adjusted to the application parallelism by considering the changing characteristics of resources. Two rescheduling policies, for suspension and migration, are also described. The ratios between the actual and the predicted execution times are measured continuously for a given application. When the average of the computed ratios is greater than the tolerance threshold, a request for migrating is sent to the rescheduler. Finally, the study also includes an overview of self-adaptive software systems.

In [14] a thorough investigation for applications adaptation and resources selection is presented. In particular, information related to resources communication and processing times is gathered periodically during applications execution. This information is used for estimating the application requirements, so that, when a resource overloads, affecting negatively in the application performance, it is replaced. The approach claims to reduce bottleneck and resources overload. Finally, in [1] a report of existing adaptive systems solutions is provided. The article also includes recommendations for enabling autonomic operations in grid systems.

The works detailed above have a common characteristic: to improve the grid infrastructures performance. Several techniques are applied for that purpose (scheduling, rescheduling, notification, etc.) but all of them are designed from a system/administration point of view. In this work, we propose a self-adaptive model for selecting resources in an efficient way by neither modifying nor controlling their behaviour (i.e. the model does not use scheduling neither allocation; it does not apply migration neither tasks replication techniques). Hence, it is expected to improve the infrastructure throughput as well as to guide applications for dealing with environmental changes.

3 Model Definition

As stated, in the present contribution we propose a mathematical formulation for obtaining the efficiency of grid infrastructures resources. This formulation combined with the *Small-world phenomenon* leads to the *ERS* model. This model

[2] `http://www.globus.org/`

[3] The *Information System* is a vital part of any grid infrastructure, providing fundamental mechanisms for discovery, monitoring, planning and thus for adapting application behaviour.

provides a self-adaptive capability to the changing properties of a grid infrastructure (as described in Section 3.3) in an autonomous way. Before proceeding to the detailed model, a brief introduction to certain grid concepts, for a better understanding, is exposed below.

3.1 Basic Grid Concepts

In a typical grid infrastructure users interact with different components through the *user interface* (*UI*). This machine is the access point to the infrastructure, so that, at this point the authentication and authorization processes come into place. Then, users send jobs to the infrastructure; these jobs are handled by the *Workload Management System* (*WMS*). This service is allocated in a machine denoted as *Resource Broker* (*RB*), and its functions are to determine the best *Computing Element* (*CE*) and to register the status and output of jobs (the *Information System* is also involved in these operations). The process used by the *RB* for selecting a *CE* is known as *match-making* and is based on availability and proximity criteria. The *CE* is a scheduler within a grid site (denoted as resources centre) that determines in which computing nodes (*Worker Node WN*) jobs will be executed.

Considering that the present approach is defined from the user point of view, the actions it performs belong to the users command set. Moreover, the resources that the model monitors, classifies and selects are the *CEs*, matching the selection criteria. In the following section the mathematical formulation of the *ERS* model is described.

3.2 The Mathematical Formulation to Measure Resources Efficiency

The *ERS* model selects the resources which best fit the application requirements by continuously monitoring their efficiency. To obtain the efficiency value two parameters are considered: the historical value ϵ_i of finished tasks[4] of the $i-th$ resource and the historical value μ_i of processing time used for these finished tasks. Both parameters are obtained for every *CE* selected during the application execution.

The ϵ_i parameter is computed as the ratio of the amount of successfully finished tasks SFt_i and the total number of assigned tasks At_i (Eq.1).

$$\epsilon_i = SFt_i/At_i \ . \tag{1}$$

Concerning μ_i, for each finished task j the processing time consumed by resource i is measured (Eq.2). The processing time $T_{i,j}$ includes both the communication time $Tcomm_{i,j}$ between the $i-th$ resource and other grid services and the computation time $Tcomp_{i,j}$ for the corresponding task j.

[4] Every task whose grid status is *Done* or *Aborted* is considered a finished task. Also tasks whose *lifetime* ends before they are fully executed are considered finished tasks. The concept of *lifetime* is introduced to avoid jobs running indefinitely, due to overload resources.

$$T_{i,j} = Tcomm_{i,j} + Tcomp_{i,j} \ . \tag{2}$$

All these $T_{i,j}$ values $\{T_{i,1}, T_{i,2}, ... T_{i,SFt_i}\}$ are used to compute the processing time average value $\bar{\chi_i}$ for the $i-th$ *CE* (Eq.3) up to the number of finished tasks SFt_i at that time.

$$\bar{\chi_i} = (\sum_{j=1}^{SFt_i} T_{i,j})/SFt_i \ . \tag{3}$$

Then, the historical value is calculated as the normalized difference of $\bar{\chi_i}$ and the *lifetime* lt fixed for performing tasks (see Eq.4).

$$\mu_i = (lt - \bar{\chi_i})/lt \ . \tag{4}$$

Finally, the efficiency value E_i is measured by using both ϵ_i and μ_i values along with two *relevance* parameters a and b (as shown in Eq.5). These parameters are introduced in the model for allowing users to specify their priorities (either success tasks rate or execution time) in the experiments.

$$E_i = (a \cdot \epsilon_i + b \cdot \mu_i)/(a + b) \ . \tag{5}$$

3.3 The Efficient Selection Model: Combining the Mathematical Formulation with the Small-World Phenomenon

As mentioned, the *ERS* model has two main goals: reducing the applications execution time and increasing the number of successfully finished tasks. It is expected that both objectives lead to an improvement in the infrastructure throughput. The model is based on the mapping between two spaces: **1)** A task space J composed by n independent and parallel tasks (they only differ in the input parameters values). **2)** A dynamic resource space R consisting on the m heterogeneous resources which are available in the corresponding grid infrastructure at execution time. This mapping is a many-to-one relationship, that is, one or more tasks could be associated to the same *CE*. As shown in Figure 1, at the beginning of application execution the model prepares a subset of J - denoted as T - and sends it to execution. The resources selected to perform this T task set compose a subset of R known as RT. At that moment, the model does not have efficiency metrics of resources, hence, the elements included in RT are chosen in a random way. The reason for launching a task set at the beginning is to foster the model to acquire efficiency metrics more rapidly (fast learning).

During the rest of the execution, the model handles the spaces as follows: when a task $t_\alpha \in J$ ends it execution the model evaluates the resource $r_\alpha \in R$ associated to t_α. Then, based on the *Small-world* property, an efficient resource is selected for a new task.

Once the rules for managing the workspaces J and R are established, the next step is to define the efficient selection process which characterized the proposed approach. As mentioned above, it is based in Watts-Strogatz small-World

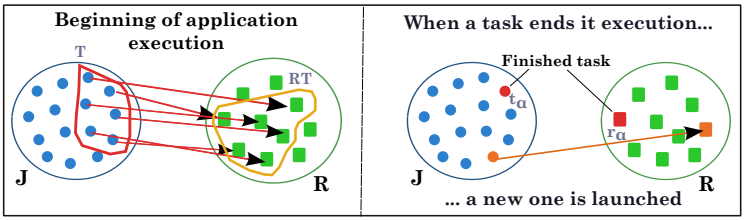


Fig. 1. Management of J and R spaces by the model during the whole application execution. The spaces are managed in a special way at the beginning of the execution.

network [9][13]. As stated in [9], a *Small-world* effect can be implemented by combining a random long-range search with a local shortcuts search. This type of algorithms are denoted as *Small World Optimization Algorithms* (*SWOA*) [3]. Hence, we apply this technique in our search process for enhancing the *ERS* proposal. The components applied to both search processes are described below.

- A workload threshold ϖ used to move from a local to a global search (i.e. from local to long-range search). For every resource its real workload[5] is monitored. The model assumes that a resource is overloaded when its workload value is equal or very close to ϖ.
- Evaluated resource set S_E: vector ordered from lowest to highest efficiency values used in the local shortcuts search. The vector is composed by those resources whose efficiency has been measured.
- Unevaluated resource set S_{UE}: vector consisting on unevaluated resources of R (unselected until that moment).

Local Shortcuts Search

The main action of this local search process is to select an efficient resource within the r_α neighbourhood. We set this neighbourhood as the two nearest neighbours of r_α (v_1 and v_2 in Figure 2), that is, resources with closest efficiency values in S_E. There are two special cases in which r_α only has one neighbour: if its efficiency value is the lowest or the highest.

When the efficiency of r_α is obtained, its value is inserted in S_E in an orderly way. Then, the model must select one of its neighbours for the next t_α. The first option is the most efficient neighbour v_1, but if it is overloaded (ϖ is applied) the model tries the second option: the other nearest neighbour v_2. If this resource is also overloaded, the model applies the random long-range search (there is a jump in the search triggered by ϖ).

[5] Both the load produced by our application and local load (derived from other external experiments which are also using this *CE*) are considered. It is a normalized value.

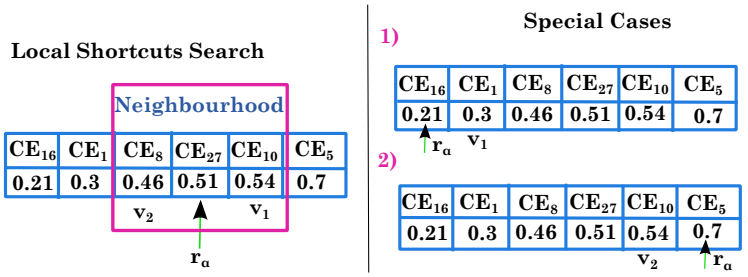


Fig. 2. The Local Shortcuts Search process defined for the *ERS* model. The resources efficiency and workload values are considered.

Random Long-range Search

Regarding the long-range search process, from the S_{UE} set a resource is randomly selected[6]. This resource must fulfil one single requirement: not to be overloaded. Therefore, this random search is repeated until a resource that meets the requirement is found. There are two special cases: on the one hand, at a certain moment all S_{UE} elements could be overloaded and, on the other hand, S_{UE} could be empty. In both cases the long-range search would be performed on the S_E set.

Summarizing the model behaviour: first a task set T is launched into execution and the corresponding tasks are monitored. When a task t_α finishes its execution, the corresponding resource r_α is evaluated. This resource is inserted into S_E in order. In this step the model applies the *Small-world* property: a local search is performed and an efficient neighbour of r_α is selected. If the two neighbours are overloaded, the algorithm jumps and starts a global random search. When the *Small-world* process ends, a new task is assigned to the resulting efficient resource[7]. These processes of monitoring, evaluation and selection are repeated until all $t_\alpha \in J$ are processed. Finally, the model registers all the information gathered during application execution in output files.

4 Performance Evaluation

As mentioned, the model evaluation is performed on a real grid infrastructure. The *ES-NGI* initiative hosts a wide range of projects, each of them with their dedicated grid environments. We are affiliated to the Ibergrid project[8] which has about 30 *CEs* (a reasonable quantity for evaluating the model).

Two scenarios have been designed to determine if the main goals are achieved: an execution time reduction and an increase of successfully finished tasks rate. In both scenarios, the *ERS-SW* is compared with the Traditional Resources

[6] $S_E \cap S_{UE} = \emptyset$ and $S_E \cup S_{UE} = R$

[7] As the resources efficiency is monitored constantly, inefficient and overloaded resources are avoided. This way, the **self-adaptive** ability is provided.

[8] `http://www.ibergrid.eu/`

Selection (*TRS*)[9] in grid computing. This selection is based on proximity and availability criteria. Finally, the testing application used in both scenarios is a fourth-order implicit approximation of the Runge-Kutta method [11].

Scenario 1

In this first scenario we evaluate the *ERS* model learning capacity and how it is influenced by the T size. The scenario is composed by 5 tests, so that, in every of them 10 experiments are performed for each version (*ERS-SW* and *TRS* respectively). The space J size is fixed to 200 tasks in all tests. The size of T ranges from 5 to 40 tasks (see Figure 3).

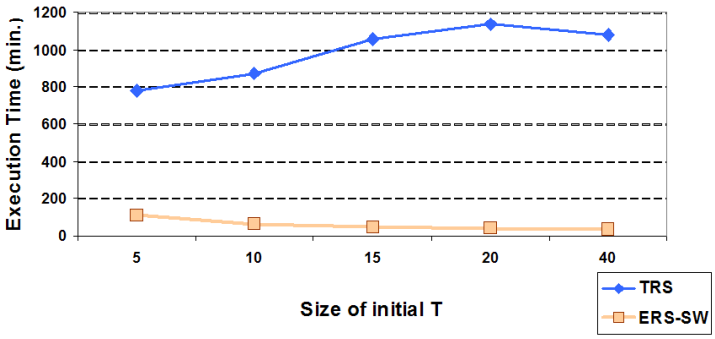


Fig. 3. Total execution time vs size. For higher size values the model learning improves.

As shown in Figure 3, *ERS-SW* reduces the execution time[10] with respect to *TRS*. It can be noticed that as T is increased, the difference between both versions grows. This means that for high T values the model finds the efficient resources faster (fast learning). Regarding the successfully finished tasks rate, *ERS-SW* version does not only improve it but also its results do not depend on the size of the initial T, as shown in Figure 4.

Scenario2

The objective of this second scenario is to determine the range of applications in which the *ERS* strategy can be applied. In this case, the size of T is set to 10 because from previous results (scenario 1) this is reported as the most appropriate minimum value for T. Six tests compose this scenario and in all of them the size of J is varied (from 50 to 500 tasks).

As in the first evaluation scenario, the *ERS-SW* improves the *TRS* results. For the two first values of space J the time difference between both versions (*ERS-SW* and TRS) is not very large. However, this difference becomes progressively

[9] A grid user normally uses this type of selection when interacts directly with a grid infrastructure. It is the standard grid selection mechanism denoted as *match-making*.

[10] The execution time in *ERS-SW* includes the application execution time and the whole model processing time.

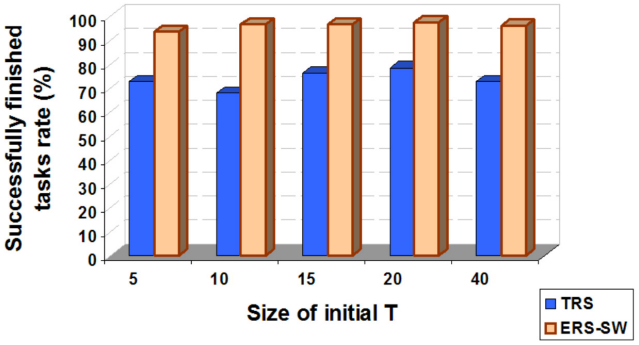


Fig. 4. Successfully finished tasks rate for the two versions. *ERS-SW* gets a uniform successfully tasks rate.

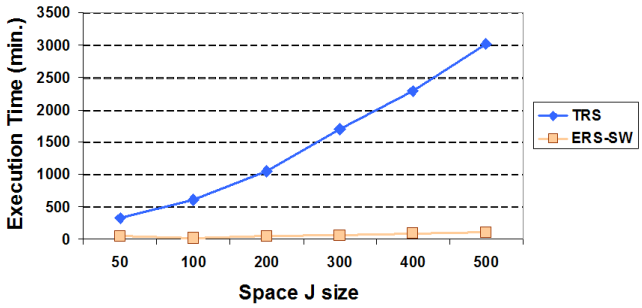


Fig. 5. Results obtained in the second scenario. The *ERS-SW* achieves better results as the size of J increases.

more significant. For example, when $J = 500$ the execution time of *TRS* is 3019 minutes (about 50 hours) while *ERS-SW* gets 108 minutes. Concerning the successfully completed tasks rate, it is improved from 70% (*TRS*) to 95% (*ERS-SW*). Finally, it can be concluded that there is a clear benefit in using the self-adaptive model proposed herein for large production applications in grid environments.

5 Conclusions

This paper proposes an efficient selection model to solve the problem of resources management in grid computing. The *ERS* model has been designed from the user point of view and it is based on the *Small-world* property. This provides a self-adaptive capability, allowing applications to deal with a changing environment. From the results obtained in the evaluation phase it is concluded that it is a feasible solution for grid users, as it increases the successfully finished tasks rate and it reduces the applications execution time. Future work involves enhancing the proposed model by applying new algorithms and to consider other grid services.

Acknowledgement. María Botón-Fernández is supported by the PhD research grant of the Spanish Ministry of Economy and Competitiveness at the Research Centre for Energy, Environment and Technology (CIEMAT). The authors would also like to acknowledge the support of the European Funds for Regional Development.

References

1. Batista, D.M., Da Fonseca, L.S.: A Survey of Self-adaptive Grids. IEEE Communications Magazine 48(7), 94–100 (2010)
2. Berman, F., Wolski, R., Casanova, H., Cirne, W., Dail, H., Faerman, M., Figueira, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, N., Su, A., Zagorodnov, D.: Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Systems 14(4), 369–382 (2003)
3. Du, H., Wu, X., Zhuang, J.: Small-World Optimization Algorithm for Function Optimization. In: Jiao, L., Wang, L., Gao, X.-b., Liu, J., Wu, F. (eds.) ICNC 2006, Part II. LNCS, vol. 4222, pp. 264–273. Springer, Heidelberg (2006)
4. Erdos, P., Rény, A.: On the Evolution of Random Graphs. Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5, 17–61 (1960)
5. Foster, I.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: Sakellariou, R., Keane, J.A., Gurd, J.R., Freeman, L. (eds.) Euro-Par 2001. LNCS, vol. 2150, pp. 1–4. Springer, Heidelberg (2001)
6. Foster, I.: What is the Grid? A three Point Checklist. GRIDtoday 1(6), 22–25 (2002)
7. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution in Grids. Software-Practice & Experience 34(7), 631–651 (2004)
8. Keung, H.N.L.C., Dyson, J.R.D., Jarvis, S.A., Nudd, G.R.: Self- Adaptive and Self-Optimising Resource Monitoring for Dynamic Grid Environments. In: DEXA 2004, Proceedings of the Database and Expert Systems Applications, 15th International Workshop, Washington DC, USA, pp. 689–693 (2004)
9. Kleinberg, J.: The Small-world Phenomenon: an Algorithm Perspective. In: Proceedings of The Thirty-second Annual ACM Symposium on Theory of Computing, Portland, OR, USA, pp. 163–170 (2000)
10. Newman, M., Barabási, A.-L., Watts, D.J.: The Structure and Dynamics of Network. Princeton University Press (2006)
11. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipies in C. Press Syndicate of the University of Cambridge, New York (1992)
12. Vadhiyar, S.S., Dongarra, J.J.: Self Adaptivity in Grid Computing. Concurrency and Computation: Practice and Experience 17(2-4), 235–257 (2005)
13. Watts, D.J., Strogatz, S.H.: Collective Dynamics of Small-world Networks. Nature 393, 440–442 (1998)
14. Wrzesinska, G., Maasen, J., Bal, H.E.: Self-adaptive Applications on the Grid. In: Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, San Jose, California, USA, pp. 121–129 (2007)

Partial Imitation Hinders Emergence of Cooperation in the Iterated Prisoner's Dilemma with Direct Reciprocity

Mathis Antony, Degang Wu, and K.Y. Szeto

Department of Physics, Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong
phszeto@ust.hk

Abstract. The evolutionary time scales for various strategies in the iterated Prisoner's Dilemma on a fully connected network are investigated for players with finite memory, using two different kinds of imitation rules: the (commonly used) traditional imitation rule where the entire meta-strategy of the role model is copied, and the partial imitation rule where only the observed subset of moves is copied. If the players can memorize the last round of the game, a sufficiently large random initial population eventually reaches a cooperative equilibrium, even in an environment with bounded rationality (noise) and high temptation. With the traditional imitation rule the time scale to cooperation increases linearly with decreasing intensity of selection (or increasing noise) in the weak selection regime, whereas partial imitation results in an exponential dependence. Populations with finite lifetimes are therefore unlikely to ever reach a cooperative state in this setting. Instead, numerical experiments show the emergence and long persistence of a phase characterized by the dominance of always defecting strategies.

1 Introduction

We use the Prisoner Dilemma [10] (PD) as an example of a two player game to study the impact of incomplete information in the imitation process. When two players play the PD game, each of them can choose to cooperate (C) or defect (D). Each player is awarded a payoff depending on his own and the opponent's move. Cooperation yields R (S) if the opponent cooperates (defects) and defection yields T (P) if the opponent cooperates (defects). R is the *Reward for cooperation*, S is the *Sucker's payoff*, T is the *Temptation to defect* and P is the *Punishment*. In the PD, $T > R > P > S$ and $2R > T + P$ to prevent collusion if the game is played repeatedly. The PD is a so called *non zero sum game* because one players loss does not equal his opponent's gain. By cooperating, both players win, by mutually defecting they both lose. In this paper we do not vary these payoff parameters but employ a set of commonly used values with high temptation: $T = 5$, $R = 3$, $P = 1$, $S = 0$. These values were also used in Axelrod's famous PD computer tournament [2]. For an excellent review of the PD literature, we refer the reader to [12].

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 92–101, 2013.
© Springer-Verlag Berlin Heidelberg 2013

The tragedy behind the PD briefly consists in the fact that the best strategy for a selfish individual (D) is the worst strategy for the society. The expectation of playing D is greater than the expectation of playing C (independent of the opponents strategy), but cooperating yields a higher total payoff. The state where no player has anything to gain by changing her own strategy (the so called *Nash Equilibrium*) occurs only when all players defect. Hence, if the players imitate the behaviour of the more successful players, defection will dominate if we do not provide any additional circumstances to encourage cooperative behaviour. Nowak et al. summarized *five rules for the emergence of cooperation* [5]: kin selection, direct reciprocity [3,4], indirect reciprocity, network reciprocity [6,13] and group selection. Network reciprocity has attracted an received particular attention recently in the light of co-evolutionary dynamics [8,9] where the network topology of the underlying interaction network evolves alongside the agent strategies.

The mechanism at work here is direct reciprocity. Players are given a memory or in other words the ability to remember a fixed number of recent outcomes of the *PD* games. Each player is then supplied with a set of answers to respond to every possible history. We call this set of moves a Strategy[1].

Players will then imitate other players by adopting their strategies. But if the strategies are elaborate an imitation may be challenging. As an illustration, assume Alice and Bob are playing chess and Alice is winning every game. Even if Bob recalls all of Alice's moves he will not be able to imitate her strategy completely until a huge number of games have been played. Instead he may attempt to improve his own strategy by adapting elements of Alice's strategy exposed to him during previous games. Intuitively the more complex a strategy, the more difficult it should be for a player to imitate it. We incorporate this condition by means of an imitative behaviour we refer to as *partial Imitation Rule (pIR)*. According to this rule a player can only imitate based on her knowledge about the opponent's strategy gathered during the most recent encounter with this opponent. This is in contrast to what we call the *traditional Imitation Rule (tIR)* that allows players to imitate the complete strategy of their opponents. Numerical experiments are performed to analyse the impact of the adjustment to the imitation behaviour. We show that it leads to new phenomena that do not occur under tIR, such as a phase dominated by defecting strategies.

The rest of this paper is structured as follows: in section 2 the terms memory, strategy and the imitation rules are defined. We also provide details about our numerical experiments. Results are presented and discussed in section 3. Our conclusions follow in section 4.

2 Methods

In this section we explain the concepts of memory and strategies and define the partial imitation rule we previously introduced in [14,15].

[1] It is common to refer to *cooperate* and *defect* as strategies. A set of rules telling the player when to cooperate or defect is then called a *meta-strategy*. For convenience we refer to the former as "moves" and to the latter as "strategies" instead.

2.1 Memory and Strategies

A player who can remember the last n rounds of the PD game has a n-step memory. We denote the ensemble of n-step strategies as M_n. The total number of strategies in M_n is $|M_n|$. As a player with one-step memory we need to remember two moves, our move and the one of our opponent. There are four possible outcomes (DD,DC, CD and CC, where the first letter is the move of the first player and the second letter is the move of the second player) of the PD game. For our one-step memory we need to have a response (either C or D) to each of these four possible outcomes. Thus our strategy can be represented by a 4-bit string where every bit is a response to one outcome of the previous round of the game. We add a bit for the first move against an unknown opponent. A strategy in M_1 is denoted as $S_0|S_{DD}S_{DC}S_{CD}S_{CC}$ where S_0 is the first move and S_{DD}, S_{DC}, S_{CD} and S_{CC} are the moves that follow DD, DC, CD and CC histories respectively. Thus there are $|M_1| = 2^5 = 32$ possible strategies as there are two choices for each S_i, either C or D. Three famous strategies are Grim-Trigger (GT): $C|DDDC$, Tit-For-Tat (TFT): $C|DCDC$ and Pavlov or "win-stay-lose-shift": $C|CDDC$.

In this paper we focus entirely on M_1. For models with longer memory horizon see for example [3,4]. There are four always defecting strategies in M_1, namely $D|DDDD$, $D|DDDC$, $D|DDCD$ and $D|DDCC$. We refer to these strategies as *all-D* type strategies. For a given opponent, they all score the same. However when using the partial imitation rule described below they can produce different child strategies. The same applies to the four always cooperating strategies. A strategy is *nice* if the first move and all moves that follow mutual cooperation are C. In M_1 those are the strategies $C|XXXC$ where X may be either C or D. A *retaliating* strategy defects after its attempt to cooperate is met with defection. In M_1 retaliating strategies are $X|XXDX$. The only four *nice* and *retaliating* strategies in M_1 are therefore *TFT*, *GT*, *Pavlov* and $C|CCDC$.

2.2 Partial Imitation Rule (pIR)

In general the *traditional Imitation Rule* (tIR) has the following simple character: a player i will imitate the strategy of player j, who is usually one of the players who interacts with i, with a certain probability given by a monotonically increasing smoothing function $g(\Delta U)$ where $\Delta U = U_j - U_i$ is the payoff difference between player i and j. For the rest of this paper we use the following smoothing function, which also introduces a temperature like noise factor K allowing for irrational choices by players. If player i has been selected to imitate player j then he will carry out the imitation with probability

$$P(i \text{ imitates } j) = g(\Delta U) = \frac{1}{1 + \exp\left(\frac{-\Delta U}{K}\right)} \tag{1}$$

We note that this probability has the form of Fermi distribution and is a step function at zero noise ($K = 0$). For low values of the noise factor K, player i imitates player j very rationally. For high values of K however, player i imitates

player j with a probability close to $1/2$ as for $K \gg |\Delta U|$ we have $P \approx \frac{1}{2} + \frac{\Delta U}{4}\frac{1}{K}$. The imitation in this case is similar to a scenario with weak selection intensity with the addition of the constant $1/2$ which introduces noise in finite sized populations.

The traditional imitation rule implicitly makes a bold assumption in the case of memory agents: the imitating player is assumed to know the entire strategy of the role model. Depending on how the two players interacted, some of the role model's strategy may be unknown to the imitator. In order to strip the players from these "mind reading" abilities we use the *partial Imitation Rule* in which the imitator only adapts the parts of the role model's strategy which have been exposed during their interaction.

We illustrate the difference between the two imitation rules with the example of a $C|DDDD$-strategist, Alice, imitating a TFT-strategist, Bob. Figure 1 shows the transition graph for this encounter. As shown in the transition graph the CD

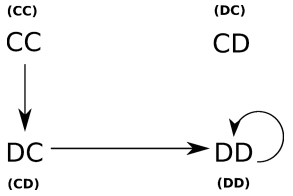


Fig. 1. Transition graph between the $C|DDDD$ player Alice and the TFT ($C|DCDC$) player Bob from Alice's point of view. Bob's point of view is described by the moves in parenthesis. The recurrent states is DD and the average recurrent state payoff is therefore P for both Alice and Bob.

state from Alice's perspective (or DC state from Bob's perspective) never occurs. Bob has therefore never used the S_{DC} move of his strategy in this encounter. According to pIR Alice cannot copy such hidden moves. Hence if the $C|DDDD$ player Alice imitates the TFT player Bob according to the partial Imitation Rule she will only imitate S_0, S_{DD}, S_{CD} and S_{CC}, hence becoming herself a GT- and not a TFT-player. If Alice imitates Bob using tIR she will copy the entire TFT strategy and become a TFT player herself. By using tIR we implicitly assume Alice has found a way to expose Bob's hidden response to the DC history.

2.3 Simulation

An important parameter in the iterated Prisoner's Dilemma with memory is the number f of rounds played during an encounter between two players i and j. If $f = 1$ we have a "one-shot" game and the agents can not make use of their memory. From our recent works in [14,15]we understand that the number f affects our results in a complex way. Here, we follow the approach employed in [3] for replicator dynamics: assuming that the number f is sufficiently large the average

payoff per instance of the PD game played in a confrontation is well approximated by the average payoff from the recurrent states of the transition graph. In other words, we address the case $f \to \infty$ in our simulations by considering only payoff accumulated in recurrent states of the transition graph.

Let U_{ij} be the average recurrent state payoff obtained by an i strategist playing against a j strategist. If N is the total number of players and every player i plays against all other players and himself, his average payoff per encounter is

$$U_i = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{S_i^T U S_j} = \boldsymbol{S_i^T U} \langle \boldsymbol{S} \rangle \ , \quad i = 1, 2 \ldots |M_n| \tag{2}$$

Where $\boldsymbol{U}$ is the $|M_n| \times |M_n|$ real Matrix with coefficients $\boldsymbol{U_{ij}} = U_{ij}$. The vector $\boldsymbol{S_i^T} = (0 \ldots 0\, 1\, 0 \ldots 0)$ is a $|M_n|$ boolean vector where the m-th entry is equal to 1 only if player i is an m-strategist. $\langle \boldsymbol{S} \rangle$ is the $|M_n|$ dimensional real column-vector of the "average strategy" that can also be written as $\langle \boldsymbol{S} \rangle_m = \rho_m$ where $m = 1, 2, \ldots, |M_n|$ and ρ_m is the number density of m-strategists. Note that the summation of ρ_m over all m equals 1.

Initially every player is assigned a random strategy out of the 32 strategies in M_1. The system is then evolved with random sequential updating from time $t = 0$ until some final time t_f. We chose two players i and j and let them play against all opponents and themselves to accumulate an average payoff per encounter U_i and U_j. By using the reasoning above this can be achieved by randomly selecting two strategies i and j with probabilities equal to ρ_i and ρ_j respectively and evaluating U_i and U_j according to equation 2. Agent i then imitates agent j with a probability given by equation 1 according to pIR or tIR. If this imitation occurs we adjust ρ_i and ρ_k where k is the children strategy produced by the imitation process. The outline of this procedure is given in algorithm 1.

Algorithm 1. Outline of simulation procedure

```
chose N strategies randomly from a uniform distribution
compute ⟨S⟩
for t = 0 to t_f do
  for n = 1 to N do
    pick random strategy i and j with probability ρ_i and ρ_j respectively
    compute U_i and U_j
    i imitates j according to tIR or pIR with probability g(ΔU)
    ρ_i ← ρ_i − 1/N
    ρ_k ← ρ_k + 1/N
  end for
end for
```

A Monte Carlo sweep, generation or one time unit corresponds to N such updates. As a result of introducing noise, the strategy fractions during a typical simulation fluctuate considerably even if N is large. The fate of the entire

population is then subject to the survival of a few key strategies, such as GT, in the early phase of evolution. As our primary interest is neither directed towards these special cases of evolution nor towards finite size effects we use a very large number of players. We have found that by choosing $N = 2.5 \cdot 10^7$ we can obtain reliable data for noise factors up to at least $K = 150$, which is sufficient for our observations. Based on our experiments we may state here that if the aforementioned extinctions of key strategies due to random fluctuations do not occur, the strategy fractions as a function of time for smaller population sizes are very similar to those we will observe below.

3 Results

In this section we first discuss a typical simulation at high noise in section 3.1 for illustrative purposes and to introduce the *all-D* phase. In section 3.2 we report and discuss the time scale of the evolution of cooperation in our model.

3.1 All-D Phase

The number density, concentration or fraction of a strategy i in a population of players is denoted as ρ_i. Figure 2 shows the number density of key strategies as a function of time for a typical pIR simulation with high noise factor (here $K = 100$). We notice that in a first phase the $D|DDCD$ and $D|DDCC$ fraction increase rapidly but die out soon thereafter to make room for the $D|DDDD$ and $D|DDDC$ strategy. These two strategies dominate for a long time but the $C|DDDD$ and GT fraction increase progressively up to a point where all four remaining strategies are about equally abundant. In figure 2 this occurs around $t = 9500$. The GT fraction then rises rapidly while the other strategies die out. Eventually we are left in an equilibrium state where all players cooperate by using the GT strategy.

We first give a intuitive explanation of these observations. When noise is high, the imitation probability is close to 1/2, thus, the imitation processes occur in many directions with only marginal drift towards imitation of better strategies. We denote the process of an A-strategist becoming a C-strategist by imitating a B-strategist as $A \xrightarrow{B} C$. If the C-strategist may turn back into an A-strategist by imitating the A-strategist we say that this imitation is directly reversible.

In an early tumultuous phase the majority of strategies die out rapidly. The famous and well scoring strategies TFT and *Pavlov* do not survive this phase either. Due to the nature of pIR there is a net drift away from these strategies at $K = 100$. In this early period of evolution, the players will obtain the highest payoff by exploiting the naive players in the initial random setup and adopt the *all-D* type strategies. As a result, most players go out of this early extinction phase as $D|DDCD$ and $D|DDCC$ defectors. The fate of these two strategies is governed by $D|DDCD \xrightarrow{GT \text{ or } C|DDDD} C|DDDD$ and $D|DDCC \xrightarrow{GT \text{or } C|DDDD} GT$. Note that these imitation processes are not directly reversible under pIR. To illustrate

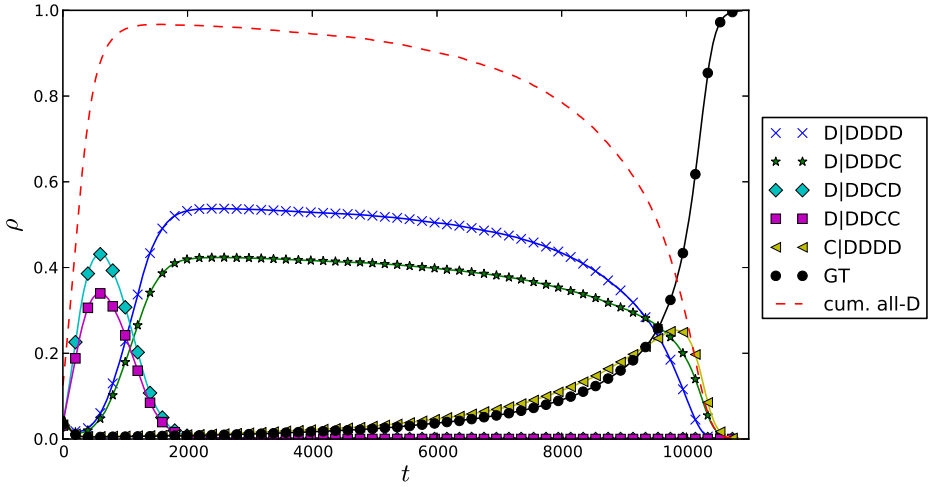


Fig. 2. Strategy fractions during a typical simulation at high noise, $K = 100$. The *cumulative all-D fraction* is given by $\rho_{\text{cum all-D}} = \rho_{D|DDDD} + \rho_{D|DDDC} + \rho_{D|DDCD} + \rho_{D|DDCC}$.

this interesting phenomena, note that we have $GT \xrightarrow{D|DDCC} D|DDDC$. This means that once a $D|DDCC$-strategist has imitated a GT-strategist, he may not turn back into a $D|DDCC$-strategist simply by imitating the $D|DDCC$ strategy. As a result, the $D|DDCD$ and $D|DDCC$-strategists are gradually converted into $D|DDDD$ and $D|DDDC$ strategists by the interaction with GT and $C|DDDD$ players. The extinction of $D|DDCD$ and $D|DDCC$ is the inevitable consequence.

The main imitation processes during the following long phase of dominance of the $D|DDDD$ and $D|DDDC$ strategies are $D|DDDD \xrightarrow{GT \text{ or } C|DDDD} C|DDDD$, $D|DDDC \xrightarrow{GT \text{ or } C|DDDD} GT$ and $C|DDDD \xrightarrow{GT} GT$. As all these processes are directly reversible it takes a very long time for the players to drift towards the better scoring GT strategy. GT is the only strategy that scores higher than the other three remaining key strategies as GT players cooperate with GT players but defect against all the other remaining strategies.

3.2 Time Scale for Emergence of Cooperation

From extensive simulations we know that all populations eventually reach an equilibrium state[2] in which more than half the players use GT. We use this fact

[2] Note that for tIR and pIR and all considered values of the noise factor K only *nice* strategies exist in the equilibrium state. In this state we have completely random drift among the surviving strategies for tIR. For pIR on the other hand there is no such drift as *nice* strategies do not change by imitating other nice strategies.

to analyse the time scale of our numerical experiments and define a quantity called the *GT First Passage Time* τ_{GT}, which is the time at which the population contains more GT players than defectors for the first time or in other words the *GT First Passage Time* is the earliest time t such that $\rho_{GT} > \rho_{\text{cum. all-D}}$. Where $\rho_{\text{cum. all-D}} = \rho_{D|DDDD} + \rho_{D|DDDC} + \rho_{D|DDCD} + \rho_{D|DDCC}$ is the *cumulative all-D fraction*. We can use this definition for the *GT First Passage Time* τ_{GT} as an indicator of the time scale to equilibrium of tIR and pIR populations. The first passage times τ_{GT} as function of K are shown in figure 3.

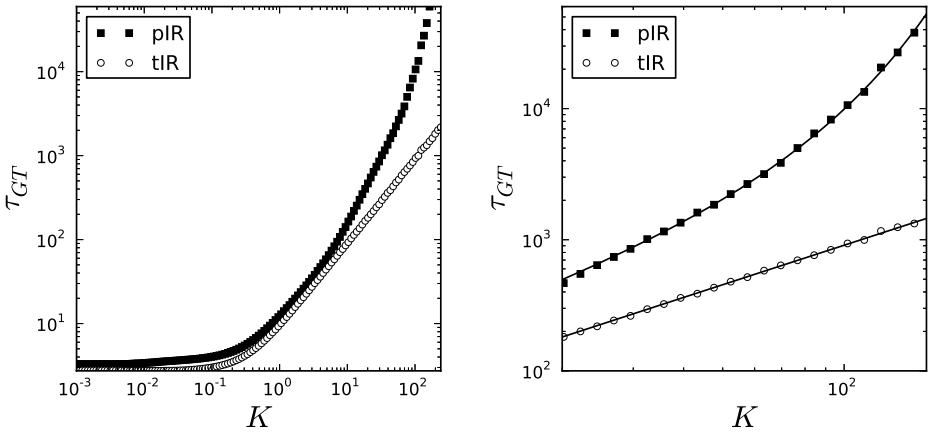


Fig. 3. First passage time τ_{GT} for $\rho_{GT} > \rho_{c.all-D}$ for both imitation rules. For illustrative purpose the data is shown in two figures, focusing on the higher end of the noise factor scale on right hand side. The figure on the right hand side also shows a linear and exponential fit for tIR and pIR respectively. Note the logarithmic scale on both axes on the left and linear scale on the x-axis and logarithmic scale on the y-axis on the right.

We first examine the figure on the left hand side. For both imitation rules τ_{GT} increases monotonically with K. For $K < 0.1$ τ_{GT} is practically constant for tIR and increases only marginally with K for pIR. For tIR the growth is linear for $K \gtrsim 0.5$. Between $0.5 \lesssim K \lesssim 5$, τ_{GT} also exhibits a linear relationship with K for pIR. However, by examining the figure on the right hand side we observe an exponential increase of τ_{GT} for pIR and $K > 40$.

The exact rate at which τ_{GT} increases with K depends on the values of the payoff parameters R, T, S and P. We note in passing that an exponential relationship for high values of the noise factor K is not unique to our choice of parameters. Notably, we also observe it for the entire range of T in the *weak Prisoner's Dilemma* [7] with $R = 1$, $1 < T < 2$ and $S = P = 0$.

In summary we make two important observations: the time scale to cooperation increases with the noise factor K for both imitation rules and in the high noise regime the relation is linear with K for tIR and exponential with K for

pIR. The importance of random fluctuations increases with K. This makes the emergence of cooperation less certain in finite sized populations for higher values of K. Furthermore, due to the exponential growth of τ_{GT} for pIR and because the lifetimes of real populations are finite the cooperative equilibrium might never be reached[3]. The partial imitation rule thus introduces a new obstacle for the emergence of cooperation.

4 Conclusion

By performing numerical experiments on a large population of prisoners with finite memory playing the iterated Prisoner's Dilemma game on a fully connected network we have shown that a direct and fast route to cooperation may not exist unless the players are given the ability to copy unknown parts of their role models' strategies. Our adjustment to the imitative behaviour shows that incomplete information about opponent strategies has important consequences for the emergence of cooperation in such a society of prisoners. If the information about the wealth of opponents is vague (at high noise or in the weak selection regime) the majority of prisoners stick to defecting strategies for a very long time. As we have seen that in this environment the duration of this route to cooperation scales exponentially with the noise factor, we must question the significance of such a cooperative equilibrium for populations with finite life times.

The *partial Imitation Rule* also presents a new challenge for famous *PD* strategies. The *Grim Trigger* strategy has a fundamental advantage over other nice strategies such as *TFT* and *Pavlov*, despite the very similar performance of all these strategies: *GT* is the only strategy that is easy to imitate for always defecting strategies. Or in other words, we observe that all other things being equal, the easiest strategy to adopt is the most successful strategy.

Many of the problems involving evolutionary games and memory could be reexamined under the new light of partial information. Although we have shown that considering partial rather than complete information has a strong impact already in the case of a one-step memory, we expect even more drastic effects in the case of longer memory. A natural extension of our work will therefore include more intelligent prisoners with longer memory. We expect that the simple *GT* strategy will not be as efficient anymore because once defecting it does not provide a way to reestablish cooperation.

Finally, finite size effects as well as the topology of the underlying network is an interesting topic for further investigation. In two dimensions [14,15], we also reported striking difference for the two imitation rules considered here. We may therefore extend our studies to other networks, such as *scale-free networks* which model the topology of real societies more accurately [1].

Acknowledgements. We thank the three anonymous referees for their valuable comments. K.Y. Szeto acknowledges the support of grant FSGRF13SC25.

[3] For instance, an estimated 99.9% of all species that ever lived on earth have become extinct [11].

References

1. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74(1), 47–97 (2002)
2. Axelrod, R.: The Evolution of Cooperation. Basic Books (1984)
3. Baek, S.K., Kim, B.J.: Intelligent tit-for-tat in the iterated prisoner's dilemma game. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 78(1), 011125 (2008)
4. Lindgren, K., Nordahl, M.G.: Evolutionary dynamics of spatial games. Physica D Nonlinear Phenomena 75, 292–309 (1994)
5. Nowak, M.A.: Five rules for the evolution of cooperation. Science 314(5805), 1560–1563 (2006)
6. Nowak, M.A., May, R.M.: The spatial dilemmas of evolution. Int. J. of Bifurcation and Chaos 3(1), 35–78 (1993)
7. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359(6398), 826–829 (1992)
8. Pacheco, J.M., Traulsen, A., Nowak, M.A.: Coevolution of strategy and structure in complex networks with dynamical linking. Phys. Rev. Lett. 97, 258103 (2006)
9. Perc, M., Szolnoki, A.: Coevolutionary games–a mini review. Biosystems 99(2), 109–125 (2010)
10. Poundstone, W.: Prisoner's Dilemma: John Von Neumann, Game Theory and the Puzzle of the Bomb. Doubleday, New York (1992)
11. Raup, D.M.: Extinction: Bad genes or bad luck? W.W. Norton, New York (1991)
12. Szabo, G., Fath, G.: Evolutionary games on graphs. Physics Reports 446(4-6), 97–216 (2007)
13. Szabo, G., Vukov, J., Szolnoki, A.: Phase diagrams for an evolutionary prisoner's dilemma game on two-dimensional lattices. Phys. Rev. E 72(4), 047107 (2005)
14. Wu, D., Antony, M., Szeto, K.Y.: Evolution of Grim Trigger in Prisoner Dilemma Game with Partial Imitation. 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, Part I. LNCS, vol. 6024, pp. 151–160. Springer, Heidelberg (2010)
15. Wu, D., Antony, M., Szeto, K.Y.: Partial Imitation Rule in Iterated Prisoner Dilemma Game on a Square Lattice. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. SCI, vol. 284, pp. 141–150. Springer, Heidelberg (2010)

A Memetic Approach to Bayesian Network Structure Learning

Alberto Tonda[1], Evelyne Lutton[2],
Giovanni Squillero[3], and Pierre-Henri Wuillemin[4]

[1] INRA UMR 782 GMPA, 1 Av. Brétignières, 78850, Thiverval-Grignon, France
alberto.tonda@grignon.inra.fr
[2] INRIA Saclay-Ile-de-France, AVIZ team, Bâtiment 650, 91405, Orsay Cedex, France
evelyne.lutton@grignon.inria.fr
[3] Politecnico di Torino, DAUIN, Corso Duca degli Abruzzi 124, 10129, Torino, Italy
giovanni.squillero@polito.it
[4] LIP6 - Département DÉSIR, 4, place Jussieu, 75005, Paris
pierre-henri.wuillemin@lip6.fr

Abstract. Bayesian networks are graphical statistical models that represent inference between data. For their effectiveness and versatility, they are widely adopted to represent knowledge in different domains. Several research lines address the NP-hard problem of Bayesian network structure learning starting from data: over the years, the machine learning community delivered effective heuristics, while different Evolutionary Algorithms have been devised to tackle this complex problem. This paper presents a Memetic Algorithm for Bayesian network structure learning, that combines the exploratory power of an Evolutionary Algorithm with the speed of local search. Experimental results show that the proposed approach is able to outperform state-of-the-art heuristics on two well-studied benchmarks.

Keywords: Memetic Algorithms, Evolutionary Algorithms, Local Optimization, Bayesian Networks, Model Learning.

1 Introduction

Bayesian networks are probabilistic graphical models that represent a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). They are widely used to encode knowledge and perform predictions in many different fields, ranging from medicine to document classification, from computational biology to law.

It is theoretically possible to learn the optimal structure for a Bayesian network from a dataset. However, the number of possible structures is superexponential in the number of variables of the model [1] and the problem of Bayesian network learning is proved to be NP-hard [2].

The machine learning community delivered fast heuristic algorithms that build the structure of a Bayesian network on the basis of conditional independence

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 102–111, 2013.
© Springer-Verlag Berlin Heidelberg 2013

evaluations between variables [3] [4]. On the other hand, several attempts have been made in evolutionary computation to tackle this complex issue [5] [6] [7]. Interestingly, many evolutionary approaches also feature local search techniques to improve the quality of the results.

This paper presents a memetic approach to Bayesian network structure learning. The proposed technique exploits an evolutionary framework evolving initial conditions for a state-of-the-art heuristic that efficiently explores the search space. The fitness function is based on the Akaike information criterion, a metric taking into account both the accuracy and the complexity of a candidate model.

An additional objective of this work is to link the community facing the complex Bayesian network structure learning problem, to the community of memetic computing. While combinations of heuristics and evolutionary optimization are prominently featured in the literature related to structure learning, to the authors' knowledge the methods presented are almost never ascribed to the field of memetic algorithms. In the authors' opinion, an explicit interaction between the two communities could lead to extremely beneficial results.

2 Background

In order to introduce the scope of the present work, some necessary concepts about Bayesian networks and memetic algorithms are summarized in the following.

2.1 Bayesian Networks

A Bayesian Network (BN) is defined as a *graph-based model of a joint multivariate probability distribution that captures properties of conditional independence between variables* [8]. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. The network could thus be used to compute the probabilities of the presence of various diseases, given the symptoms.

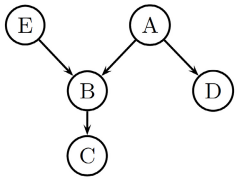
Formally, a Bayesian network is a directed acyclic graph (DAG) whose nodes represent variables, and whose arcs encode conditional dependencies between the variables. This graph is called the *structure* of the network and the nodes containing probabilistic information are called the *parameters* of the network. Figure 1 reports an example of a Bayesian network.

The set of parent nodes of a node X_i is denoted by $pa(X_i)$. In a Bayesian network, the joint probability distribution of the node values can be written as the product of the local probability distribution of each node and its parents:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i))$$

2.2 The Structure Learning Problem

Learning the structure of a Bayesian network starting from a dataset is proved to be a NP-hard problem [2]. The algorithmic approaches devised to solve this



Node	Parents	Probabilities
A		$P(A=a_1) = 0.99$ $P(A=a_2) = 0.01$
B	A,E	$P(B=b_1 \mid A=a_1, E=e_1) = 0.5$ $P(B=b_2 \mid A=a_1, E=e_1) = 0.5$ $P(B=b_1 \mid A=a_1, E=e_2) = 0.1$ $P(B=b_2 \mid A=a_1, E=e_2) = 0.9$ $P(B=b_1 \mid A=a_2, E=e_1) = 0.4$ $P(B=b_2 \mid A=a_2, E=e_1) = 0.6$ $P(B=b_1 \mid A=a_2, E=e_2) = 0.2$ $P(B=b_2 \mid A=a_2, E=e_2) = 0.8$

Node	Parents	Probabilities
C	B	$P(C=c_1 \mid B=b_1) = 0.3$ $P(C=c_2 \mid B=b_1) = 0.7$ $P(C=c_1 \mid B=b_2) = 0.5$ $P(C=c_2 \mid B=b_2) = 0.5$
D	A	$P(D=d_1 \mid A=a_1) = 0.8$ $P(D=d_2 \mid A=a_1) = 0.2$ $P(D=d_1 \mid A=a_2) = 0.7$ $P(D=d_2 \mid A=a_2) = 0.3$
E		$P(A=e_1) = 0.75$ $P(A=e_2) = 0.25$

Fig. 1. On the left, a directed acyclic graph. On the right, the parameters it is associated with. Together they form a Bayesian network BN whose joint probability distribution is $P(BN) = P(A)P(B|A,E)P(C|B)P(D|A)P(E)$.

problem can be divided into two main branches: score-and-search meta-heuristics and algorithms that rely upon statistical considerations on the learning set.

Evolutionary Approaches. Among score-and-search meta-heuristics, evolutionary algorithms are prominently featured. Several attempts to tackle the problem have been tested, ranging from evolutionary programming [6], to cooperative coevolution [5], to island models [7].

Interestingly, some of the evolutionary approaches to Bayesian network structure learning in literature already show features of memetic algorithms, hinting that injecting expert knowledge might be necessary to obtain good results on such a complex problem. For example, [6] employs a *knowledge-guided mutation* that performs a local search to find the most interesting arc to add or remove. In [9], a local search is used to select the best way to break a loop in a non-valid individual. The K2GA algorithm [10] exploits a genetic algorithm to navigate the space of possible node orderings, and then runs the greedy local optimization K2, that quickly converges on good structures starting from a given sorting of the variables in the problem.

Dependency Analysis Algorithms. Dependency analysis algorithms are a class of heuristics that build Bayesian network structures from data through an evaluation of the conditional independence between variables. They are able to deliver results of high quality in negligible time, even if they suffer from the classical issues of greedy algorithms, such as being trapped into local optima.

One of the best algorithms in this category is known as *Greedy Thick Thinning* (GTT) [3]. Starting from a completely connected graph, first GTT applies the well-known PC algorithm [11], that cuts arcs on the basis of conditional independence tests; then, it starts first adding and then removing arcs, scoring the network after each modification and using a set of heuristic metrics to avoid a premature convergence.

Bayesian Search (BS) [4] is another state-of-the-art heuristic in the same group. Unlike GTT, BS is not deterministic: it makes use of a given number of

restarts from random sparse network layouts, finally returning the best network to the user.

Both GTT and BS implementations can be found in products such as GeNie/SMILE [12].

2.3 Memetic Algorithms

Memetic algorithms are *population-based metaheuristics composed of an evolutionary framework and a set of local search algorithms which are activated within the generation cycle of the external framework.* [13]. First presented in [14], they gained increasing popularity in the last years [15].

The main attractiveness of these stochastic optimization techniques lies in their ability of finding quickly high-quality results, but still maintaining the exploration potential of a classic evolutionary algorithm. Their effectiveness is proven in several real-world problems [16] [17].

3 Proposed Approach

Trying to reap the benefits of both evolutionary algorithms (efficient exploration, resistance to local optima attraction) and human-devised heuristics (speed, efficient exploitation of small parts of the search space), a memetic algorithm is applied to the complex problem of Bayesian network structure learning. The algorithm evolves an initial Bayesian network that will be then optimized by a state-of-the-art dependency analysis algorithm. In this first approach, the local search heuristic is applied to every individual.

The main novelties introduced by the proposed approach are the local optimization used, the state-of-the-art GTT heuristic, and the evolutionary algorithm's individual representation, expressing a set of forced and forbidden links of arbitrary size.

The framework structure is summarized in Figure 2.

3.1 Evolutionary Framework

The evolutionary core chosen for the experiments is an open-source EA [18]. Each individual represents a set of initial conditions, forbidden and forced arcs that have to appear inside the final network created by the local search. Every condition in the set follows the syntax:

`<forbidden/forced> <starting node> <end node>`

The genome has a minimum length of 1 condition, and no maximum length. It is interesting to notice that there is no *apriori* control on repetitions, or contradictory conditions (e.g., forbid *and* force the arc between A and B). Each group of repeated conditions is considered only once, while individuals with contradictions are discarded with a low fitness value. Finally, individuals whose condition enforce the local search to produce an invalid Bayesian network (e.g., forcing an arc from A to B and an arc from B to A would create a graph with a cycle) are discarded during fitness evaluation.

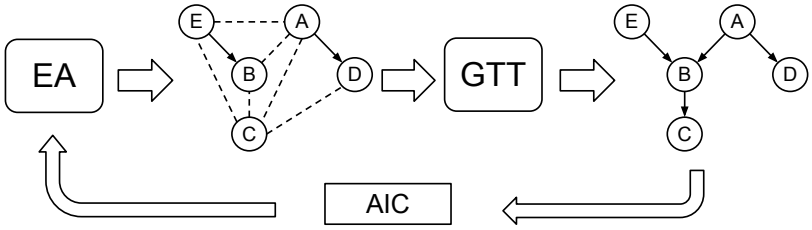


Fig. 2. Structure of the proposed approach. The evolutionary algorithm creates starting conditions where some arcs are forced (e.g. arc from E to B and from A to D) and some arcs are forbidden (e.g. arc from B to D and arc from E to D). The local search performed by GTT returns a complete structure, compliant with the conditions, manipulating freely the unconstrained arcs. The final structure is evaluated, and its AIC score is used as fitness by the evolutionary algorithm.

3.2 Fitness Function

The Akaike information criterion (AIC) is a measure of the relative goodness of fit of a statistical model [19]. It is grounded in the concept of information entropy, in effect offering a relative measure of the information lost when a given model is used to describe reality. It can be said to describe the trade-off between bias and variance in model construction, or loosely speaking, between accuracy and dimension of the model. Given a data set, several candidate models may be ranked according to their AIC values: thus, AIC can be exploited as a metric for model selection.

When dealing with Bayesian networks, AIC is expressed as a composition of the loglikelihood, a measure of how well the candidate model fits the given dataset, and a penalty tied to the dimension of the model itself. The dimensional penalty is included because, on the one hand, the loglikelihood of a Bayesian network usually grows monotonically with the number of arcs, but on the other hand, an excessively complex network cannot be validated or even interpreted by a human expert. The loglikelihood of a model M given a dataset T is computed as

$$LL(M|T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} log_2 \frac{N_{ijk}}{N_{ij}}$$

where n is the number of variables, q_i is the total number of possible configurations the parent set $pa(X_i)$ of the stochastic variable X_i, r_i is the number of different values that variable X_i can assume, N_{ijk} is the number of instances in the dataset T where the variable X_i takes its k-th value x_{ik} and the variables in $pa(X_i)$ take their j-th configuration w_{ij}, and N_{ij} is the number of instances in the dataset T where the variables in $pa(X_i)$ take their j-th configuration w_{ij}.

Taking for example the Bayesian network BN described in Figure 1, the loglikelihood of a dataset composed of one sample such as $T = (a_1, b_2, c_1, d_2, e_2)$ would be equal to

$$\begin{aligned} LL(BN|T) = log_2(P(A = a_1) \cdot P(B = b_2|A = a_1, E = e_2) \cdot \\ \cdot P(C = c_1|B = b_2) \cdot P(D = d_2|A = a_1) \cdot P(E = e_2)) = \\ = log_2(0.99 \cdot 0.9 \cdot 0.5 \cdot 0.2 \cdot 0.25) = -5.49 \end{aligned}$$

It is important to notice that datasets are usually composed by multiple samples, and that the final loglikelihood is the sum of the loglikelihoods of each sample. Using the same formulation, the dimensional penalty of model M can be expressed as

$$|M| = \sum_{i=1}^{n} (r_i - 1)q_i$$

In the canonical representation, the final AIC score is expressed as:

$$AIC = -2 \cdot (LL - |M|)$$

AIC is to be minimized.

4 Experimental Setup

The effectiveness of the proposed approach is compared against GTT and BS. First, the memetic algorithm is run with a stagnation condition: if the best individual in the population remains the same for 10 generations, the algorithm stops. Then, the total number of evaluations performed is used as a reference to compare its performance against the two other approaches. BS is assigned an equal number of restarts; for the deterministic GTT, an equal number of starting random configurations are generated, following the same initialization procedure of the memetic algorithm.

In all the experimental evaluations, the algorithm has a population size μ=30, an offspring size[1] λ=30, a stagnation stop condition of 10 generations, and a set of operators that can collectively alter, remove or add a condition from an individual, and cross over two individuals, with one or two cut points. Individuals are chosen for reproduction with a tournament selection scheme. The strength and the activation probability of the genetic operators, as well as the size of the tournament selection, are self-adapted during the evolutionary run.

GTT and BS use default settings[2] with the exception of the maximum number of parents for each node, set to 10. GTT makes use of K2 as the type of priors, while BS has a probability 0.1 of an arc appearing in a random restart, a prior link probability 0.001 and a prior sample size 50.

When GTT is run as local search in the proposed memetic algorithm, it makes use of the same settings.

[1] In the chosen evolutionary framework, λ represents the number of genetic operators applied at each generation. Since some of the operators, most notably crossovers, can produce more than one child individual, the number of individuals actually generated at each step fluctuates between 30 and 60, with values often falling around 45.

[2] Default settings for the heuristics provided by the SMILE [12] framework, see `http://genie.sis.pitt.edu/wiki/SMILearn`

5 Experimental Results

Two widely studied Bayesian network benchmarks are chosen for the experiments: ALARM [20], that features 37 variables and 42 arcs; and INSURANCE [21], that features 27 variables and 52 arcs. For each of the considered Bayesian network benchmarks, three datasets with 500, 1,000 and 10,000 samples respectively are created. The structure learning algorithms are executed 10 times for each dataset, each run with a given number of restarts/evaluations.

The Bayesian networks reconstructed by each algorithm are compared on three different metrics: loglikelihood, dimension and overfitting. Loglikelihood expresses the adherence of the given model to the training data. The dimension of the model is a measure of its complexity, where simpler models are preferred to more complex ones from a human perspective. Finally, the overfitting on the training data is evaluated by computing the loglikelihood of the candidate solution on a validation set of unseen data, composed of 5,000 samples.

Results are summarized in Table 1 and Table 2, with a highlight in Figures 3 and 4.

Table 1. Results for the three algorithms on the considered datasets of the Bayesian network ALARM. Results in **bold** are the best, on the basis of a two-sample Kolmogorov-Smirnov test with $p < 0.05$; when two distributions are indistinguishable but better than the third, they are highlighted in ***bold italics***.

alarm-500						
Methodology	Dimension	StDev	Loglikelihood	StDev	Overfitting	StDev
Original	509.00	-	-7,510.47	-	-75,195.1	-
(trained on dataset)	509.00	-	-7,588.84	-	-77,989.3	-
GTT (1,900 restarts)	***464.90***	11.30	-7,673.69	16.35	-79,618.9	84.41
BS (1,900 restarts)	1,298.90	87.34	-7,896.71	143.34	-85,260.0	1,781.41
Memetic Algorithm	***463.20***	28.44	**-7,629.34**	33.02	**-79,118.8**	451.57
alarm-1,000						
Original	509.00	-	-15,023.0	-	-75,195.1	-
(trained on dataset)	509.00	-	-15,045.8	-	-76,919.9	-
GTT (1,400 restarts)	564.70	19.78	-15,097.2	21.74	-77,659.3	187.28
BS (1,400 restarts)	1,546.20	149.36	-15,808.8	130.65	-83,381.4	680.57
Memetic Algorithm	**537.40**	35.80	**-15,057.7**	24.58	**-77,438.3**	236.46
alarm-10,000						
Original	509.00	-	-150,099	-	-75,195.1	-
(trained on dataset)	509.00	-	-149,993	-	-75,357.6	-
GTT (1,300 restarts)	779.00	40.57	-150,088	73.55	-75,506.8	31.79
BS (1,300 restarts)	3,369.90	553.5	-156,690	940.70	-79,550.9	447.24
Memetic Algorithm	**674.00**	53.80	**-150,026**	27.92	**-75,433.7**	26.96

6 Results Discussion

The proposed approach is proved to outperform state-of-the-art heuristic techniques for the considered metrics on all the datasets, providing networks with smaller dimension, higher loglikelihood. There are, however, open research questions raised by the analyses, that is worth addressing separately.

Table 2. Results for the three algorithms on the considered datasets of the Bayesian network INSURANCE. Results in **bold** are the best, on the basis of a two-sample Kolmogorov-Smirnov test with $p < 0.05$; when two distributions are indistinguishable but better than the third, they are highlighted in ***bold italics***.

insurance-500						
Methodology	Dimension	StDev	Loglikelihood	StDev	Overfitting	StDev
Original	1,008.00	-	-9,337.06	-	-94,354.2	-
(trained on dataset)	1,008.00	-	-9,678.63	-	-101,884	-
GTT (1,700 restarts)	497.90	29.86	-9,598.68	14.67	-100,792	133.40
BS (1,700 restarts)	706.30	95.64	-9,668.27	74.27	-102,599	915.88
Memetic Algorithm	**458.60**	9.60	**-9,562.40**	9.86	**-100,278**	132.36
insurance-1,000						
Original	1,008.00	-	-19,024.7	-	-94,354.2	-
(trained on dataset)	1,008.00	-	-19,335.0	-	-98,346.4	-
GTT (1,600 restarts)	673.60	62.08	***-19,357.9***	51.16	-98,166.9	177.04
BS (1,600 restarts)	1,020.10	142.30	-19,606.9	196.80	-100,324	1,022.27
Memetic Algorithm	**574.80**	50.64	***-19,323.3***	52.60	**-97,884.2**	234.84
insurance-10,000						
Original	1,008.00	-	-187,858	-	-94,354.2	-
(trained on dataset)	1,008.00	-	-188,070	-	-95,194.7	-
GTT (2,000 restarts)	1,090.50	113.40	-188,274	96.98	-94,929.5	37.94
BS (2,000 restarts)	2,063.70	480.24	-192,121	883.16	-97,215.5	407.24
Memetic Algorithm	**882.00**	59.80	**-188,155**	57.76	**-94,834.0**	33.75

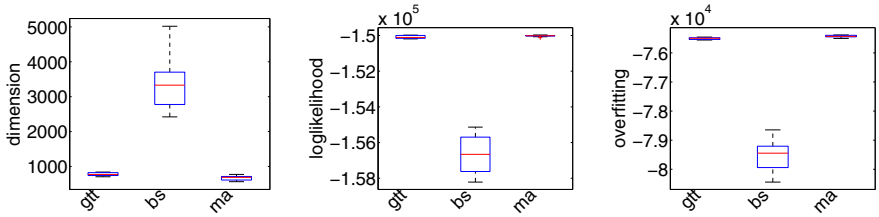


Fig. 3. Boxplots for the 10,000-sample dataset of ALARM network, for dimension, loglikelihood and overfitting

The memetic framework presented is still relatively rough. Since the local search is applied to all individuals, the very same problem can be also expressed as finding the best set of initial conditions for the heuristic optimization algorithm. The authors see the current work as a first step, and are currently working on an extension of the framework to include other state-of-the-art optimization heuristics, such as BS, in order to give more freedom to the memetic algorithm.

The exact meaning of the initial conditions used in the framework is an interesting point of reflection. At a first glance, they might be simply considered as the point from which the heuristic will start its local search. The reality, however, is more complex: arcs forbidden and arcs forced in the initial conditions cannot be altered by the local search. This significantly changes the search space, providing the heuristic not only with a starting point, but also with a set of hard constraints that cannot be altered, and that will limit the exploration to a restricted part of the original search space. Further experiments are needed to fully understand the repercussions of forbidden and forced arcs on the adopted heuristics.

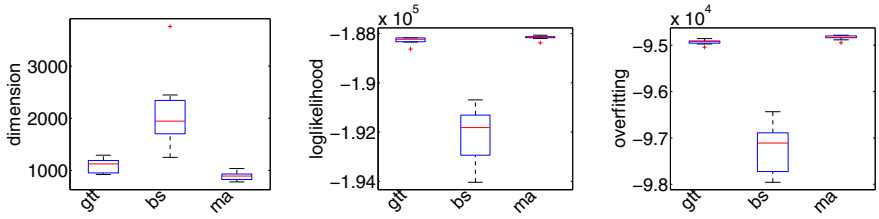


Fig. 4. Boxplots for the 10,000-sample dataset of INSURANCE network, for dimension, loglikelihood and overfitting

For a final remark on time, the heuristics with restarts and the proposed approach operate in the same order of magnitude, ranging from a few minutes for small training sets to a little more than an hour for larger ones, on the same machine.

7 Conclusions and Future Works

This paper proposes a memetic algorithm for Bayesian network structure learning, coupling the speed of local search with the exploration ability of an evolutionary algorithm. The algorithm evolves the initial conditions of a network, forcing and forbidding some of the arcs, and letting a local search manipulate the remaining connections from that starting point. Experimental results show that the approach is significantly more effective than a set of random restarts of state-of-the-art heuristic algorithms.

Future works will assimilate different structure learning heuristics in the memetic framework, embedding inside the genome of each individual additional information about the type and settings of the local search to apply.

References

1. Robinson, R.: Counting unlabeled acyclic digraphs. In: Little, C. (ed.) Combinatorial Mathematics V. Lecture Notes in Mathematics, vol. 622, pp. 28–43. Springer, Heidelberg (1977), 10.1007/BFb0069178
2. Chickering, D.M., Geiger, D., Heckerman, D.: Learning bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, USA (November 1994)
3. Cheng, J., Bell, D.A., Liu, W.: An algorithm for bayesian belief network construction from data. In: Proceedings of AI & STAT 1997, pp. 83–90 (1997)
4. Cooper, G.F., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347 (1992), 10.1007/BF00994110
5. Barriere, O., Lutton, E., Wuillemin, P.H.: Bayesian network structure learning using cooperative coevolution. In: Genetic and Evolutionary Computation Conference, GECCO 2009 (2009)

6. Wong, M.L., Lam, W., Leung, K.S.: Using evolutionary programming and minimum description length principle for data mining of bayesian networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(2), 174–178 (1999)
7. Regnier-Coudert, O., McCall, J.: An Island Model Genetic Algorithm for Bayesian network structure learning. In: 2012 IEEE Congress on Evolutionary Computation, pp. 1–8 (June 2012)
8. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using bayesian networks to analyze expression data. In: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, RECOMB 2000, pp. 127–135. ACM, New York (2000)
9. Delaplace, A., Brouard, T., Cardot, H.: Computational intelligence and security, pp. 288–297. Springer, Heidelberg (2007)
10. Larranaga, P., Kuijpers, C., Murga, R., Yurramendi, Y.: Learning bayesian network structures by searching for the best ordering with genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 26(4), 487–493 (1996)
11. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, 2nd edn., vol. 1. The MIT Press (2001)
12. Druzdzel, M.J.: SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In: American Association for Artificial Intelligence, pp. 902–903 (1999)
13. Hart, W.E., Krasnogor, N., Smith, J.E.: Memetic Evolutionary Algorithms. In: Hart, W.E., Smith, J., Krasnogor, N. (eds.) Recent Advances in Memetic Algorithms. STUDFUZZ, vol. 166, pp. 3–27. Springer, Heidelberg (2005)
14. Norman, M., Moscato, P.: A competitive and cooperative approach to complex combinatorial search. In: Proceedings of the 20th Informatics and Operations Research Meeting, pp. 3–15 (1991)
15. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation 2, 1–14 (2012)
16. Fan, X.F., Zhu, Z., Ong, Y.S., Lu, Y.M., Shen, Z.X., Kuo, J.L.: A direct first principles study on the structure and electronic properties of $be_xzn_{1-x}o$. Applied Physics Letters 91(12), 121121 (2007)
17. Nguyen, Q.H., Ong, Y.S., Lim, M.H.: A probabilistic memetic framework. IEEE Transactions on Evolutionary Computation 13(3), 604–623 (2009)
18. Sanchez, E., Schillaci, M., Squillero, G.: Evolutionary Optimization: the uGP toolkit. Springer (2011)
19. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 19(6), 716–723 (1974)
20. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In: Second European Conference on Artificial Intelligence in Medicine, London, Great Britain, vol. 38, pp. 247–256. Springer, Berlin (1989)
21. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Machine Learning 29 (1997)

Multiobjective Evolutionary Strategy for Finding Neighbourhoods of Pareto-optimal Solutions

Ewa Gajda-Zagórska

AGH University of Science and Technology, Krakow, Poland
gajda@agh.edu.pl

Abstract. In some cases of Multiobjective Optimization problems finding Pareto optimal solutions does not give enough knowledge about the shape of the landscape, especially with multimodal problems and non-connected Pareto fronts. In this paper we present a strategy which combines a hierarchic genetic algorithm consisting of multiple populations with rank selection. This strategy aims at finding neighbourhoods of solutions by recognizing regions with high density of individuals. We compare two variants of the presented strategy on a benchmark two-criteria minimization problem.

Keywords: genetic algorithm, multiobjective optimization, clustering, hierarchic genetic strategy.

1 Introduction

The aim of this paper is to present Multiobjective Clustered Evolutionary Strategy (MCES) - a method for recognizing sets and separating neighbourhoods of the Pareto sets' non-connected parts in multimodal multiobjective problems. It was firstly introduced in [5]. We will consider a basic variant of the strategy and its new modification.

A group of evolutionary algorithms approximating solutions of multiobjective problems is called Multiobjective Optimization Evolutionary Algorithms (MOEA, for comparison of MOEAs see e.g. [17]). Usually, a MOEA aims at finding a set of Pareto-optimal solutions which may not give enough information in some cases, for example in problems with non-connected Pareto fronts. It is difficult to extract knowledge about how small perturbations affect domination among solutions from the existing algorithms. In our approach, solutions from the neighbourhood of the Pareto-set are detected and may be analysed. We will focus on a Pareto-based selection (*FFGA*) by Fonseca and Fleming [3], where an individual's rank equals to the number of solutions by which it is dominated. Pareto sets and fronts in multiobjective problems were investigated for example by Preuss, Naujoks and Rudolph in [10].

We will develop the idea of recognizing sets by clustering dense regions. Whereas in many papers (see e.g. [7]) a genetic algorithm is used as a help

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 112–121, 2013.
© Springer-Verlag Berlin Heidelberg 2013

tool in clustering, we consider a combination of the two methods in the opposite way. Genetic algorithm here is used to provide a clustering method with data set as an input. The advantages of clustering in single-objective genetic algorithms were studied by Schaefer, Adamska and Telega (CGS, see e.g. [12], [1]). For other examples of two-phase global optimization strategies see [8] and [14]. Separation and estimation of the number of basins of attraction was performed by Stoean, Preuss, Stoean and Dumitrescu in [15] and in [11].

2 Strategy

The idea of MCES strategy of detecting neighbourhoods of the Pareto sets consists of combining a genetic algorithm with rank selection and a clustering method. Among many GA we would like to distinguish those which may provide best samples for clustering. In case of single-population algorithms (like Simple Genetic Algorithm, SGA [16]), early convergence may eliminate some solutions. This behaviour may result in losing information about parts of the Pareto front. Therefore, we propose to use an algorithm having both high selection pressure and globality and which properties may be theoretically verified, called Hierarchic Genetic Strategy (HGS, see [13]).

2.1 Preliminaries

In the multiobjective optimization, we are given $k \geq 2$ objective functions

$$f_i : D \to [0, M] \subset \mathbb{R},\ M < +\infty,\ i \in \{1, \ldots, k\} \tag{1}$$

defined over a decision space D. For single-objective problems $k = 1$. We assume that all objectives shall be minimized.

Definition 1. *(Pareto dominance) For any pair $(p, q) \in D \times D$, p is said to dominate q, denoted as $p \succ q$, if and only if*

$$\forall_{i=1,\ldots,k}\ f_i(p) \leq f_i(q)\ \textit{and}\ \exists_{i=1,\ldots,k}\ f_i(p) \neq f_i(q). \tag{2}$$

The *Pareto set* $\mathcal{P}$ is the set of non-dominated elements from D and its image $f(\mathcal{P}) \subset [0, M]^k$ is called the *Pareto front*. Therefore to solve the exact problem

$$\min_{x \in D} \{f(x) = (f_1(x), \ldots, f_k(x))\} \tag{3}$$

the Pareto set $\mathcal{P} \subset D$ (or the Pareto front) is sought.

Next, we present encoding function used by genetic algorithms, $code : U \to D$, where U is a set of all genetic codes called the *genetic universum*. The encoding function introduces a grid of admissible points $D_r = code(U)$. We assume that $\#U = r < +\infty$.

The genetic algorithm solves an approximate problem

$$\min_{p \in U} \{f(code(p)) = (f_1(code(p)), \ldots, f_k(code(p))\}, \tag{4}$$

where the solution is a Pareto set $\mathcal{P}_U \subset U$.

In the algorithm we use a selection mechanism such that selection probability reaches highest values on $\mathcal{P}_U$. We may assume that $code(\mathcal{P}_U)$ is close to $\mathcal{P}$ in a sense; both sets are in metric vector space $\mathbb{R}^n$. The exact analysis of distance between $code(\mathcal{P}_U)$ and $\mathcal{P}$ is still an open problem.

2.2 Hierarchic Genetic Strategy

HGS is an algorithm producing a tree-structured set of concurrent evolutionary processes (see Figure 1). The strategy was introduced by Kołodziej and Schaefer in [13]. The structure of HGS tree changes dynamically and its depth is bounded by $m < +\infty$. We will focus on the case in which each evolutionary process is governed by SGA.

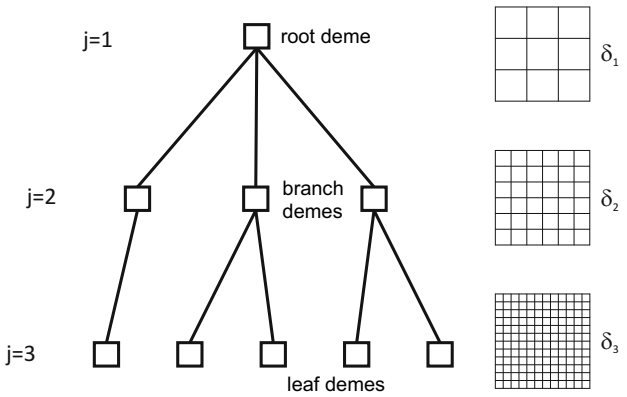


Fig. 1. HGS tree and corresponding two-dimensional meshes, $m = 3$

HGS starts with a single root deme (population), a process of the first order, performing chaotic search with low accuracy. After a fixed number of genetic epochs K called the *metaepoch* the root deme sprouts child-demes in the promising regions of the evolutionary landscape surrounding the best fitted individuals distinguished from the parental population. Child-demes perform more local search with higher accuracy. The evolution in existing demes continues in the second metaepoch, after which new demes are sprouted. Demes of order m (leaves) perform local and most accurate search. The algorithm continues until the global stop condition is reached.

HGS implements two mechanisms that prevent redundancy of the search: *conditional sprouting* and *branch reduction*. The former allows new demes to be sprouted only in regions which are not explored by sibling-demes (demes sprouted by the same parent). The latter reduces (kills) populations of the same order that perform search in the common landscape region or in already explored regions. A population is also killed when its average fitness does not change in several consecutive epochs.

Diverse search accuracies are obtained by various encoding precisions. In binary encoding, lengths of binary genotypes are different in demes at different levels. The root utilizes the shortest genotypes and leaves utilize the longest ones. To obtain search coherency for demes at various levels, a hierarchical nested encoding is used. Firstly, we define the densest mesh of phenotypes for demes of the m-th order. Afterwards, the meshes for lower order demes are recursively defined by selecting nodes from previous ones. The maximum diameter of the mesh δ_j associated with populations of the order j determines the search accuracy at this level of the HGS tree (see Figure 1). Defined mesh parameters satisfy $\delta_m < \ldots < \delta_1$.

In real-number encoding implementation of HGS, a genotype is a vector of floating point numbers. We use scaling functions to obtain a sequence of increasing genetic spaces for subsequent orders of branches. These genetic spaces are smaller for lower order branches, and spaces for the highest order branches have the genetic space of the size of the numerical representation of the given domain. For details of this representation, refer to [14].

Selection pressure is tightly connected with the probability of sampling measure in central parts of basins of attraction. It was formally proven for HGS in [6] that with certain assumptions the sampling measures spanned by the sum of leaves in HGS are sufficiently close to the sampling measure associated with the unique fixed point of the genetic operator. HGS is also effective in finding multiple local extrema (see [13]). It consists of multiple populations which explore different areas of the search space. Even when considering only highest-order demes, the algorithm performs global search and, with a small number of individuals, can cover the whole domain.

2.3 Genetic Algorithms with Heuristic

We consider genetic algorithms, from which the simplest operate on a single population being the multiset $P = (U, \eta)$ of the search space members called *individuals*. The occurrence function $\eta : U \to \mathbb{Z}_+ \cup \{0\}$ returns $\eta(i)$ which is the number of individuals with the genotype $i \in U$. The population cardinality is $\mu = \sum_{i \in U} \eta(i) < +\infty$.

The algorithm consists of producing a sequence of populations $\{P^t\}$ in the consecutive *genetic epochs* $t = 1, 2, \ldots$ starting from the population P^0 uniformly sampled from U. Mixing and selection operations depend on the algorithm, for example in case of MOEA selection is often performed with respect to the Pareto–dominance relation (see e.g. [2]).

Each finite population represented as the multiset $P = (U, \eta)$ may be identified with its frequency vector $x = \{\frac{1}{\mu}\ \eta(p)\}, p \in U$. All such vectors belong to the finite subset X_μ of the Vose simplex:

$$\Lambda^r = \left\{ x = \{x_p\};\ 0 \le x_p \le 1,\ p \in U,\ \sum_{p \in U} x_p = 1 \right\}. \tag{5}$$

An important group of algorithms which properties can be theoretically verified are *genetic algorithms with heuristic*. SGA is one of a few instances of genetic algorithms with heuristic; in SGA the probability distribution of sampling the next epoch population can be given explicitly (for details see [16]). For definition of heuristic refer to [14].

The second example of a genetic algorithm with heuristic refers to multiobjective case and was introduced in [4]. Selection operator in the presented algorithm was inspired by the Pareto-based ranking procedure FFGA where an individual's rank equals the number of solutions by which it is dominated. In the next step, population is sorted according to rank and fitness values are assigned to individuals by interpolating from the best (with the lowest rank) to the worst (with the highest rank) according to some function.

The selection operator $F : \Lambda^r \to \Lambda^r$ for the MOEA rank selection has the form:

$$F(x) = \frac{1}{x^T\ G(\Xi\ x)}\ \mathrm{diag}(x)\ G(\Xi\ x)\ , \tag{6}$$

where $x \in \Lambda^r$, $\mathrm{diag}(x)$ denotes the $r \times r$ diagonal matrix with the diagonal x, $\Xi \in \{0,1\}^r \times \{0,1\}^r$ is a *binary Pareto dominance matrix* where $\forall\ p, q \in U\ \Xi_{p,q} = 1$ if q dominates p and 0 otherwise, $G : [0,1]^r \to [0,1]^r$, $G(x)_p = g(x_p),\ p \in U$, and $g : [0,1] \to [0,1]$ is a decreasing validating function (e.g. $g(\zeta) = 1 - \zeta$).

In each MOEA epoch, selection is followed by genetic operations (e.g. mutation, crossover) which can be represented by the mixing operator $M \in C^1(\Lambda^r \to \Lambda^r)$. We do not impose any specific restrictions for this mapping. For an exemplary mixing operator see [16].

We compose selection and mixing to obtain a heuristic operator of the particular class of MOEA considered in this paper

$$\mathcal{H} = M \circ F. \tag{7}$$

It can be proven that, with certain assumptions, the sampling measure concentrates on the set of fixed points of $\mathcal{H}$ [4]. Applied rank selection causes the individuals to concentrate on the neighbourhood of Pareto-optimal solutions. Coupled with a multi-population strategy like HGS, the algorithm produces a sample ready to clustering.

2.4 Clustering

In the presented strategy, clustering is not restricted to any particular method. It is applied to recognize regions with high density of individuals. What is important, is to cluster populations that concentrate on the neighbourhoods of the Pareto-set. Leaves in HGS with applied MOEA selection scheme have this property, therefore we can use individuals from leaves as samples for clustering.

3 Example

In the following section we present two variants of the strategy, with different policies for sprouting leaves and finding clusters. These variants are illustrated

by a simple example problem in which we use two-level HGS implementation. In real-world problems more levels are usually used.

As an example we have chosen a two-criteria, two-dimensional minimization problem with the following objective functions:

$$f_1(x, y) = x \tag{8}$$

$$f_2(x, y) = g(y)(1 - \sqrt{\frac{x}{g(y)}} - \frac{x}{g(y)} \sin(10\pi x)) + 0.5 \tag{9}$$

where $g(y) = 1 + 9y$, $(x, y) \in [0, 1] \times [0, 1]$ (see Figure 2).

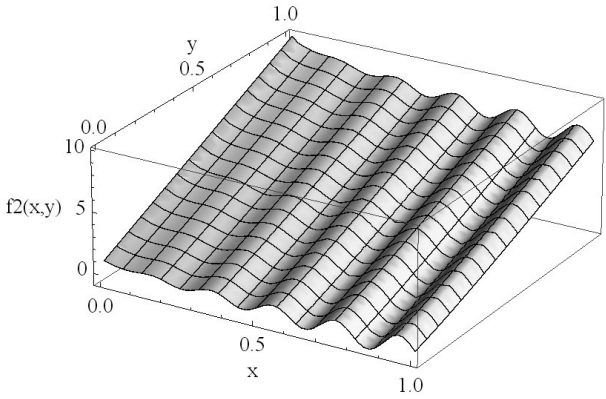


Fig. 2. Objective function f_2 (see Eq. 9)

The problem is not so easy to solve because it is multimodal and its Pareto-optimal front consists of several non-connected parts.

In the first case as a genetic engine we use a two-level real-encoding HGS with rank selection presented in the paper. Root deme consists of 50 individuals. After every 5 metaepochs non-dominated solutions are stored externally, and when the stopping condition for root is reached (20 metaepochs), leaves are sprouted from stored values. Each leaf population consists of 10 individuals and they evolve for 10 metaepochs. Parameters were set by experiments – a lower number of individuals or metaepochs resulted in unsatisfactory results (only 2 – 3 non-connected parts of Pareto set were found). From simulations with presented parameters the best run was selected.

In Figure 3 we present all individuals created by root. The individuals are quite well-spread in the entire search space and concentrate in regions with low ranks.

We obtained 30 stored points, from which leaves were sprouted and continued exploration in interesting parts of the landscape. Most of these regions are the

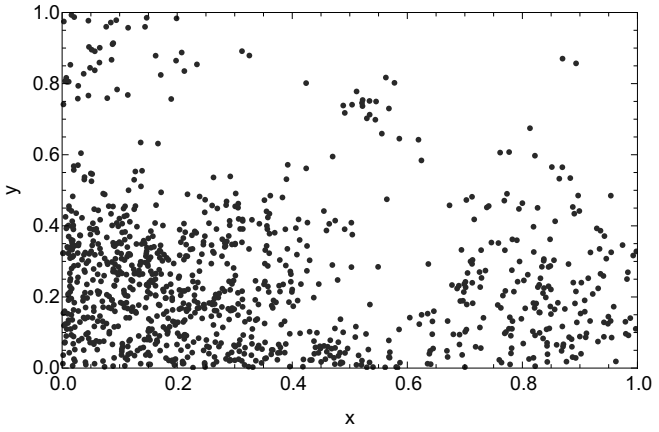


Fig. 3. Root individuals in the decision space

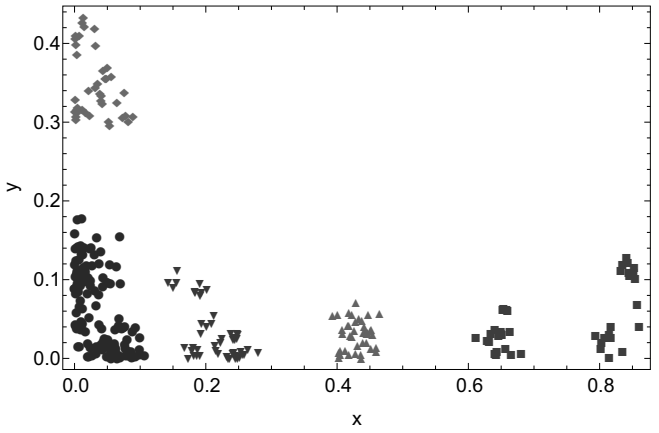


Fig. 4. Leaf individuals in the decision space. Different point markers represent clusters.

neighbourhoods of the Pareto-optimal sets. Afterwards, the results of search in leaves were clustered by k-medoids method (see e.g. [9]) with the number of clusters set to 5. In the presented example problem, found clusters represent existing parts of Pareto set quite well. The upper cluster (Figure 4) is a result of early phase of computation in root and two right-most groups of points were put in the same cluster, but these problems can be avoided by adding constraints to stored individuals and changing the assumed number of clusters.

The second case differs from the first one in several ways. Firstly, we change sprouting policy to parameterized conditional sprouting performed after each metaepoch - leaves are created only in interesting areas that were not yet explored. Secondly, it is possible to reduce the number of metaepochs (in the presented example to 15 in total). Thirdly, we treat each leaf separately instead of

clustering a set of all leaf individuals. Exemplary obtained results are presented in Figure 5. The first two groups of points overlap but all non-connected parts of Pareto set were found and there were no false solutions.

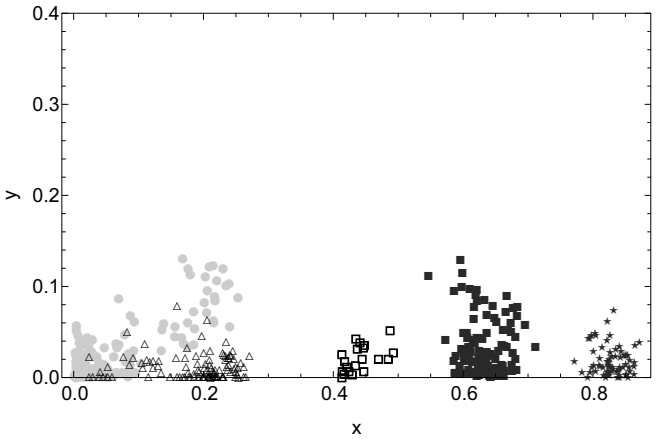


Fig. 5. Leaf individuals. Different point markers represent individuals from different leaves.

Results from the second case can be clustered with the number of clusters, which has to be provided to many clustering methods, related to number of leaves but we can also omit this last phase of MCES and assume that each leaf represents a part of the Pareto set. A comparison of average values of individuals in the first and second case, as well as the differences between them, are presented in Table 1. The upper cluster from the first case is omitted and for the two rightmost leaves (Fig. 5), which equivalents were merged into one cluster (Fig. 4), we firstly calculate the average value of both leaves together and next, calculate the distance to the cluster.

Table 1. Average values of clusters (in the first case) and leaves (in the second case) and euclidan distances between them

Case 1		Case 2		
x	y	x	y	Distance
0.036	0.066	0.065	0.029	0.047
0.215	0.031	0.186	0.012	0.035
0.43	0.031	0.441	0.021	0.014
0.739	0.048	0.632	0.032	0.024
		0.829	0.020	

Both variants of MCES found similar results for the example problem, even two overlapping leaves represent the neighbourhoods of the Pareto set non-connected parts well. The distances between the average values of clusters in case 1 and leaves in case 2 are all below 0.05.

The computational cost for the first case was higher than for the second one. Firstly, to maintain a similar quality of results (all non-connected parts of the Pareto front found) 5 more metaepochs were necessary. Secondly, there were 30 leaf populations in the first case and only 5 in the second one. In total, there were 1000 evaluations of root individuals and 3000 of leaf individuals in the first case and, respectively, 750 and 470 in the second case. If we assume that the evaluation time is the same for root and leaf individuals (which is true for the example), we obtain results 3.3 times faster in the second case. Usually, in more complex problems, the evaluation of higher order demes is much more costly than for root, therefore the difference could be even more noticeable.

There are several factors that were not included in the above estimation of computational cost. In the first case, the computational cost of the clustering phase should be added. In the second case, proper parameter tuning is necessary. It may be performed basing on some knowledge about the problem or by multiple short runs before the main calculations.

Concluding, in this section we presented an example which illustrates how two variants of MCES may be used in practice. The strategy may be successfully applied to multimodal problems and gives a better insight into the shape of a problem landscape.

4 Conclusions and Future Research

- The presented strategy of solving multiobjective optimization problem gives additional knowledge about the shape of the evolutionary landscape. What is more, it copes with multimodal problems without losing local solutions.
- Both variants of MCES can find all non-connected parts of the Pareto set. The one utilizing conditional sprouting has lower computational cost and does not require post-processing of the genetic sample. On the other hand, it requires parameter tuning to prevent repetitive leaves to be sprouted. Variant including clustering has higher computational cost but does not require any additional knowledge about the problem to be properly tuned.
- In future papers, we plan to develop the theorem allowing for theoretical verification of MCES, investigate its applicability to other types of problems and compare with different MOEAs.

References

1. Adamska, K.: Genetic Clustering as a Parallel Algorithm for Approximating Basins of Attraction. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2004. LNCS, vol. 3019, pp. 536–543. Springer, Heidelberg (2004)

2. Coello Coello, C.A., Lamont, G.B.: Applications of Multi-objective Evolutionary Algorithms. World Scientific (2004)
3. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Gen. Alg.: Proc. of the Fifth Int. Conf., pp. 416–423 (1993)
4. Gajda, E., Schaefer, R., Smołka, M.: Evolutionary Multiobjective Optimization Algorithm as a Markov System. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 617–626. Springer, Heidelberg (2010)
5. Gajda-Zagórska, E.: Recognizing sets in evolutionary multiobjective optimization. Journal of Telecommunications and Information Technology 1, 74–82 (2012)
6. Kolodziej, J.: Modelling Hierarchical Genetic Strategy as a Family of Markov Chains. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2001. LNCS, vol. 2328, pp. 595–598. Springer, Heidelberg (2002)
7. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recognition 33(9), 1455–1465 (2000)
8. Pardalos, P.M., Romeijn, H.E. (eds.): Handbook of Global Optimization, vol. 2. Kluwer (1995)
9. Park, H., Jun, C.: A simple and fast algorithm for k-medoids clustering. Expert Systems with Applications 36(2, Part 2), 3336–3341 (2009)
10. Preuß, M., Naujoks, B., Rudolph, G.: Pareto Set and EMOA Behavior for Simple Multimodal Multiobjective Functions. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 513–522. Springer, Heidelberg (2006)
11. Stoean, C., Stoean, R., Preuss, M.: Approximating the number of attraction basins of a function by means of clustering and evolutionary algorithms. In: Tandareanu, N. (ed.) 8th Int. Conf. on AIDC. Res. No. in AIDC, pp. 171–180. Reprograph Press (2008)
12. Schaefer, R., Adamska, K., Telega, H.: Clustered genetic search in continuous landscape exploration. Engineering Applications of Artificial Intelligence 17(4), 407–416 (2004)
13. Schaefer, R., Kolodziej, J.: Genetic search reinforced by the population hierarchy. In: Poli, R., De Jong, K.A., Rowe, J.E. (eds.) Foundations of Genetic Algorithms 7, pp. 383–388. Morgan Kaufmann (2003)
14. Schaefer, R., Telega, H.: Foundation of Global Genetic Optimization. Springer (2007)
15. Stoean, C., Preuß, M., Stoean, R., Dumitrescu, D.: EA-Powered Basin Number Estimation by Means of Preservation and Exploration. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 569–578. Springer, Heidelberg (2008)
16. Vose, M.D.: The Simple Genetic Algorithm. MIT Press (1999)
17. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, ETH Zurich, Switzerland (1999)

Genetic Programming-Based Model Output Statistics for Short-Range Temperature Prediction

Kisung Seo[1], Byeongyong Hyeon[1], Soohwan Hyun[2], and Younghee Lee[3]

[1] Dept. of Electronic Engineering, Seokyeong University, Seoul, Korea
[2] Hyundai Heavy Industries Research Institute, Yongin, Korea
[3] National Institute of Meteorological Research/Korea Meteorological Administration, Seoul, Korea

Abstract. This paper introduces GP (Genetic Programming) based robust compensation technique for temperature prediction in short-range. MOS (Model Output Statistics) is a statistical technique that corrects the systematic errors of the model. Development of an efficient MOS is very important, but most of MOS are based on the idea of relating model forecasts to observations through a linear regression. Therefore it is hard to manage complex and irregular natures of the prediction. In order to solve the problem, a nonlinear and symbolic regression method using GP is suggested as the first attempt. The purpose of this study is to evaluate the accuracy of the estimation by GP based nonlinear MOS for the 3 days temperatures for Korean regions. This method is then compared to the UM model and shows superior results. The training period of summer in 2007-2009 is used, and the data of 2010 summer is adopted for verification.

Keywords: temperature forecast, MOS, UM, KLAPS, genetic programming.

1 Introduction

As numerical techniques and computational power have steadily improved, various scales and types of numerical weather prediction models are becoming the most powerful tools in the quantitative forecasting of various weather elements such as temperature, precipitation, and snowfall [6,13]. The Unified Model(UM) [10] has been adopted by the Korean Meteorological Administration(KMA) as an operational model since 2007 [13].

It is well known that forecasts of numerical weather prediction models have certain defects that can be removed by statistically postprocessing their output [2,9]. Forecast models do not reliably determine weather conditions near ground level, therefore a compensation technique is required to enhance an accuracy of prediction outputs for numerical models [2,6,9,12,13]. Different statistical procedures are used in meteorology for adjusting the estimates of air temperature obtained by numeric forecasting models of climate [1-3,5,8,9,11,12].

MOS(Model Output Statistics) is an statistical technique that corrects the systematic errors of the model. A couple of indices (temperature, relative humidity,

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 122–131, 2013.
© Springer-Verlag Berlin Heidelberg 2013

wind speed and wind direction) are expected to be improved by the MOS, compared to the UM forecast alone [4,5]. The linear regression methods have been still widely used in those systems [12]. The MOS currently used in KMA for short range prediction of temperature have adopted a linear regression too.

However, a linear regression is not adequate to represent non-linear behavior of predictor variables. Moreover this approach requires pre-selected parameters to construct a regression model also, but it is difficult to decide what parameters should be included in the regression model to get best results even though some correlations can be analyzed. Therefore it has fundamental limitations to manage highly complex nature of weather predictions. Some neural network based approach [11] can represent nonlinear behaviors of model but it still requires the pre-defined variables.

To overcome these problems of existing approaches, we have proposed a seemingly more efficient approach that optimizes a compensation model for temperature predictions through nonlinear combinations of potential predictors using GP(Genetic Programming) [7]. GP based nonlinear regression can effectively search open-ended space for order and coefficient of equations. It is also powerful mean to generate open-ended high order equations and complex nonlinear forms using transcendental functions. This allows it to solve the limitations of search for a fixed linear regression model.

In this paper, a generation technique of nonlinear regression model for MOS using Genetic Programming is proposed as a first attempt as far as we know. GP based symbolic regression approach is used to perform a nonlinear regression for correcting(or compensating) a temperature prediction model.

This paper is organized as follows. Section 2 introduces a notion of numerical weather prediction. Section 3 describes genetic programming based method for non-linear MOS. Section 4 presents experimental results of temperature forecast for Korean regions by the proposed GP_MOS method, and Section 5 concludes the paper.

2 Numerical Weather Prediction

2.1 Unified Model and KLAPS

The UM(Unified Model) [10] is a numerical weather prediction and climate modeling software suite originally developed by the United Kingdom Met Office, and now both used and further developed by many weather-forecasting agencies around the world.

The Korea Meteorological Administration have an operational 12km resolution global forecasting system utilizing the Unified Model. The UM is run twice a day (00 and 12 UTC) producing forecasts from 6 h to 66 h at a 3 h interval. KLAPS(Korea Local Analysis and Prediction System) was designed to provide a computationally efficient method of combining all available sources of meteorological information into three dimensional depiction of the atmospheric state [6].

15 kinds of total 49 potential predictors were employed in our work including temperature, humidity, wind speed and accumulated rainfall as shown in Table 1.

Table 1. Potential Predictors

Types	*Potential Predictors*
Air Temperature	TS, T8, T7, T5
Thickness	DZ18, DZ17, DZ85
Dew-point	TDD8, TDD7, TDD5
Specific humidity	QS, Q8, Q7, Q5
Difference between specific humidity and saturated specific humidity at 500 hPa	DQ85, DQ75
Relative humidity	RH8, RH7, RH5
Layer averaged RH	MRH17, MRH15, MRH85
Zonal wind	US, U8, U7, U5
Meridional wind	VS, V8, V7, V5
Wind speed	WSS, WS8, WS7, WS5
Wind direction	WDS, WD8, WD7, WD5
North-westerly wind speed	NWS, NW8, NW7, NW5
North-easterly wind speed	NES, NES8, NES7, NE5
Lapse rate	LR87, LR85
Total rain amount(3hr accumulated)	PCP

2.2 MOS (Model Output Statistics)

The MOS technique aims at correcting current forecasts based on statistical information gathered from past forecasts. Not only MOS can illustrate systematic errors of the prediction model, but produce a prediction closed to average of samples. In its most popular form, it is based on a linear relation between the reference variables that we want to predict a set of model predictors at a certain lead time [2,9,12].

The MOS currently used in KMA for short term prediction of temperature have adopted a similar linear regression with equation (1). It consists of linear combination of potential predictors. ΔTS is a compensated amount for the corrected forecast.

$$\Delta TS = a_1 VAR_1 + a_2 VAR_2 + \cdots + a_N VAR_N \tag{1}$$

where, VAR_i represents one of potential predictors in Table 1.

One of problems is that the number and kind of predictors for above MOS should be fixed in advance throughout the process of a pre-selection. However, it is difficult to decide what predictors should be included in the regression model to get best results, because it is hard to analyze exact correlations among around 50 predictors for huge number of locations and periods. The other is that above linear regression is inappropriate to model non-linear relationships between a temperature prediction and predictor variables Therefore these two drawbacks hinder to build compensation model precisely.

3 Genetic Programming Based Compensation Technique

3.1 A Proposed Genetic Programming Approach

Genetic programming [7] is an extension of the genetic algorithm, using evolution to optimize actual computer programs or algorithms to solve some task, such as automated synthesis. Genetic programming can manipulate variable-sized entities and can be used to "grow" trees that specify increasingly complex predictors. An example of GP MOS regression by GP tree is shown in Figure 1. Compared to equation (1) of the linear regression, GP based MOS can express nonlinearity and periodicity much more flexibly involving multiplication, division and sinusoidal functions.

Therefore it is possible to generate open-ended high order equations and complex nonlinear forms using a tree structure and allows to solve the limitations of a linear regression approach

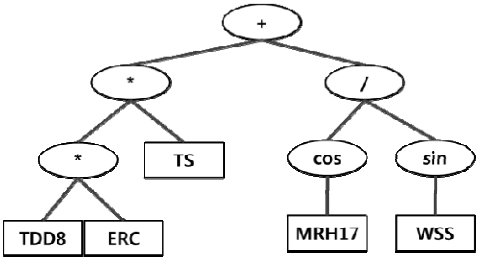


Fig. 1. Example of GP tree

Especially the fundamental problem of preselection for potential predictors can be naturally solved in GP based approach, because dominant predictors are extracted automatically through the evolution process of genetic programming. That means all candidates of predictors are considered without excluding some potential predictors in advance, therefore the possibility of optimized selection of potential predictors is much better than predetermined predictors.

Every solution of GP based MOS doesn't necessarily have same predictors, because not only the size and shape of GP tree for optimized solutions are different but also selected predictors are varied for each solution. Therefore we can generate a tailor-made compensation equation for various locations in wide range of period which have different characteristics.

3.2 Function and Terminal Sets

The function set for the proposed GP-based MOS involves 6 arithmetic operators and terminal set includes 49 potential predictors and one ERC(ephemeral random constants) as follows.

Function = {+, *, -, /, cosine, sine}
Terminal = {49 variables, ERC}

The set of primitive functions should be sufficient to allow solution of the problem at hand, but there are typically many possible choices of sets of operators that meet this condition. Through preliminary experiments, the function set above is selected.

3.3 Fitness Function

Fitness function of GP-MOS prediction is defined to minimize RMSE(Root Mean Square Error) for temperature prediction between KLAPS reference data and obtained forecast data by GP based compensation technique. It is described in equation (2), where TS_i is the surface temperature obtained by UM.

$$fitness = \sqrt{\frac{\sum_{i=1}^{Days}(KLAPS_i - GP\_MOS_i)}{Days}} \qquad (2)$$
$$GP\_MOS_i = TS_i + \Delta TS_i\_byGP$$

4 Experiments

4.1 GP Parameters

The GP programs were run on a Intel Core I7 960 3.4GHz with 4GB RAM using lil-gp [14]. The GP parameters used for the GP_MOS evolution were as follows:

Population sizes: 300
Max generation: 300
Initial Tree Depth: 2-5
Initial Tree Method: Half and Half
Max Depth: 9
Crossover Rate: 0.9
Mutation Rate: 0.1

Learning experiments are repeated 20 times from June 2007 to August 2009 for entire stations for Korea with 00 UTC(Coordinated Universal Time) and 12 UTC. A test is executed from June 2010 to August 2010 for same environment.

4.2 Experimental Results

The comparison results of average RMSE in learning experiments for 00 UTC forecast between UM and GP_MOS are shown in Figure 2 and 3. The numeric results represent RMSE between UM and GP-MOS comparing with KLAPS reference data.

The wave pattern of results by time occurs periodically. RMSE values of daytime (+06h, +27h~30h, +51h~54h) are higher than others. The average RMSE of UM is 2.035 and GP is 1.512 in learning stage, showing an improvement of 25.7%. In verification process, the average RMSE of UM is 2.059 and the result of GP is 1.565, 24.0% is improved. We observe that GP-based MOS yielded superior average RMSE to UM for both cases.

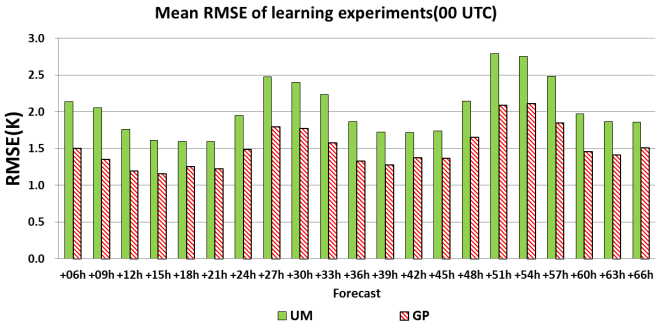


Fig. 2. Comparison of average RMSE in learning for 00UTC

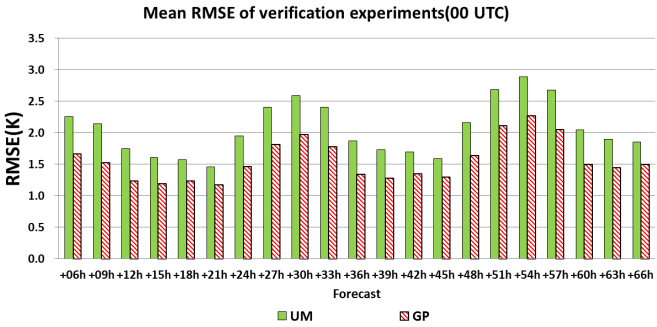


Fig. 3. Comparison of average RMSE in verification for 00UTC

Similar experimental results for 12 UTC are shown in Figure 4 and 5. The similar periodic wave pattern by time occurs but it is shifted by 12 hours. GP-based MOS showed superior average RMSE to UM for both cases like 00 UTC. GP shows again an improvement, of 24.5% in the learning stage and of 23.0% in the verification phase.

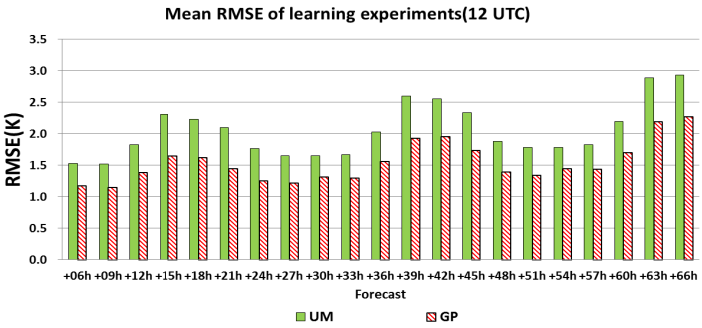


Fig. 4. Comparison of average RMSE in learning for 12UTC

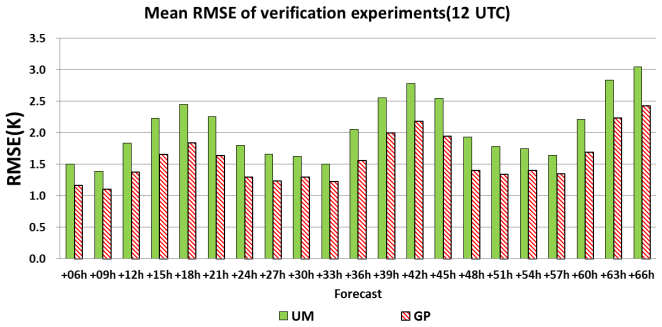


Fig. 5. Comparison of average RMSE in verification for 12UTC

Table 2. Summary of comparisons for RMSE between UM and GP_MOS

		RMSE(K) in learning (summer 2007-2009)			RMSE(K) in validation (summer 2010)		
		min	average	max	min	average	max
00 UTC	UM	0.863	2.035	7.822	0.718	2.059	7.416
	GP_MOS	0.732	1.512	3.514	0.673	1.565	3.344
12 UTC	UM	0.830	2.049	7.775	0.659	2.066	7.403
	GP_MOS	0.591	1.547	3.705	0.643	1.591	3.585

The summary of comparisons for above results from Figure 2 to 5 are shown in Table 2. We do not report on a direct comparison between our proposed GP_MOS method and the linear MOS in KMA, because that information for used predictors is not available. Rather we do compare our results indirectly with results by other researchers [5] where, a couple of MOS methods including the most popular UMOS(Updateable MOS) were tested. The result of RMSE by UMOS is 2.115 for summer in 2007. And the best RMSE result is 1.77 which is obtained by MOS_R [5]. Therefore it is certain our GP based MOS approach is quite competitive than the linear based MOS used in KMA.

4.3 An Example of GP Based Temperature Compensation in Specific Location

One of example solutions of GP based approach for 051.107 location(near Hongsung, west area of South Korea) is shown in below as a LISP expression. It is represented form of GP tree with infix notation, which has 63 nodes and depth 9, consists of sine, +, -, and * operators. Function cosine and division(/) are not selected in evolution. Especially, a series of Layer averaged RH and Lapse rate predictors are chosen several times.

GP_TREE: { sin(sin(((WS8+WS8-TS) * (LR87-RH7)) + sin(KI - sin(TS))) + sin(US + DZ18 - sin(V8))) } + { sin(sin(sin(LR87-MRH85+LR87-MRH15))) } + { sin(sin(LR87-MRH85+DZ17-MRH15+DZ17-MRH15)) } + { sin(sin(sin(sin(sin(V7))))) } - sin(sin(TS)) - U7

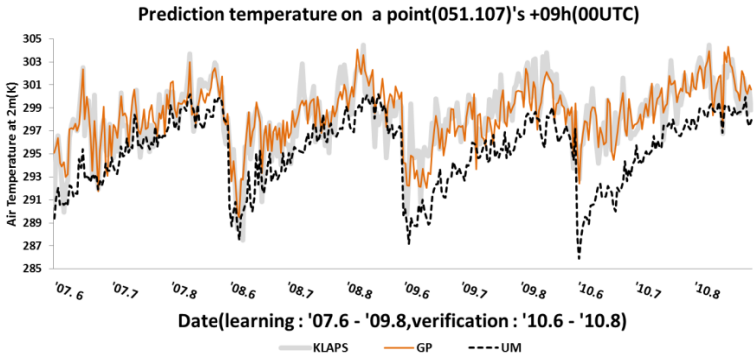


Fig. 6. Forecast temperature of +09h, 00UTC in 051.107 location(Hongsung)

Figure 6 shows obtained prediction results of period (summer 2007 ~ 2010) for 051.107 with +09h(00UTC). The gray(brighter one) line represents KLAPS observation temperature, thin dot line means forecast by GP_MOS and thick dot line indicates a prediction by UM. Learning process was executed in June 2007 ~ August 2009 period, average RMSE of UM is 3.587 and of GP_MOS is 1.478. Therefore it shows 58.8% improvement than UM. Validation process was executed in June 2010 ~ August 2010, the result of GP_MOS is improved 72.01 % than UM.

4.4 RMSE Distribution of UM and GP

RMSE of temperature forecast distributions of entire Korean region for each method are shown in Figure 7 and 8. In Figure 7, the RMSE distributions of UM for 00 UTC are displayed in left and the results of GP based method are right. Red color represents larger RMSE and blue indicates smaller RMSE. We can see the results of the proposed GP approach are better than UM. Similar results for 12 UTC are shown in Figure 8.

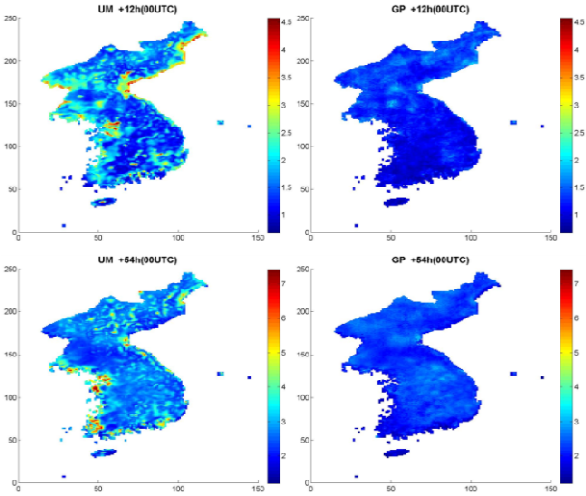


Fig. 7. RMSE distributions for UM and GP in 00 UTC

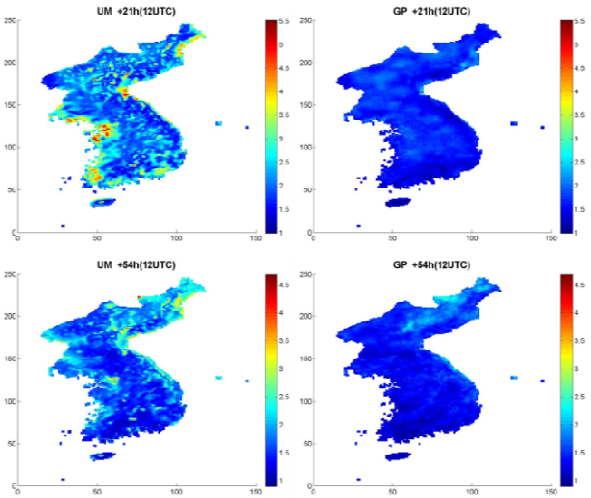


Fig. 8. RMSE distributions for UM and GP in 12 UTC

5 Conclusions

In order to improve for temperature prediction, a new nonlinear compensation technique, based on symbolic regression using Genetic Programming, is proposed and compared to UM. Experiments are executed for 9,291stations with 21 intervals and two UTCs. Learning is performed in period of June 2007 ~ August 2009 and validation is processed in summer 2010. The GP method shows superior results in average RMSE for both time period and regions compared to UM. It becomes clear that the proposed GP based method is quite competitive than the result of linear based MOS used in KMA [5] also. Further study will aim at refinement of the predictor selection and extension of the advanced GP search.

Acknowledgements. This work was supported by Korean Meteorological Administration Research and Development Program 2012(NIMR 2012-B-1)

References

1. Carvalho, J.R.P., Assad, E.D., Pinto, H.S.: Kalman filter and correction of the temperatures estimated by PRECIS model. Atmospheric Research 102, 218–226 (2011)
2. Glahn, H.R., Lowry, D.A.: The use of model output statistics (MOS) in objective weather forecasting. J. Appl. Meteor. 11, 1203–1211 (1972)
3. Glahn, B., Gilbert, K., Cosgrove, R., Ruth, D.P., Sheets, K.: The gridding of MOS. Weather and Forecasting 24, 520–529 (2009)
4. Homleid, M.: Weather dependent statistical adaption of 2 meter temperature forecasts using regression methods and Kalman filter. met. no report, Norwegian Meteorological Institute (2004)

5. Kang, J., Suh, M., Hong, K., Kim, C.: Development of updateable Model Output Statistics (UMOS) System for Air Temperature over South Korea. Asia-Pacific Journal of Atmospheric Sciences 47, 199–211 (2011)
6. Kim, Y., Park, O., Hwang, S.: Realtime Operation of the Korea Local Analysis and Prediction System at METRI. Asia-Pacific Journal of Atmospheric Sciences 38, 1–10 (2002)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge (1992)
8. Lee, Y.H., Park, S.K., Chang, D.-E.: Parameter estimation using the genetic algorithm and its impact on quantitative precipitation forecast. Annales Geophysicae 24, 3185–3189 (2006)
9. Termonia, P., Deckmyn, A.: Model-inspired predictors for model output statistics. Mon. Wea. Rev. 135, 3496–3505 (2007)
10. United Kingdom Met Office, http://www.metoffice.gov.uk
11. Ustaoglu, B., Cigizoglu, H.K., Karaca, M.: Forecast of daily mean, maximum and minimum temperature time series by three artificial neural network methods. Meteorological Applications 15, 431–445 (2008)
12. Vannitsem, S.: Dynamical Properties of MOS Forecasts: Analysis of the ECMWF Operational Forecasting System. Weather and Forecasting 23, 1032–1043 (2008)
13. Yu, X., Park, S., Lee, K., Ahn, K., Choo, S.: The gridding of MOS for high resolution forecasting. In: The Fifth Korea-Japan-China Joint Conference on Meteorology, pp. 18–21 (2011)
14. Zongker, D., Punch, B.: Lil-GP User's Manual, Michigan State University (1995)

Evolutionary Multi-Agent System in Hard Benchmark Continuous Optimisation

Sebastian Pisarski, Adam Rugała, Aleksander Byrski, and Marek Kisiel-Dorohinicki

AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
{pisarski,rugala}@student.agh.edu.pl, {olekb,doroh}@agh.edu.pl

Abstract. It turns out that hybridizing agent-based paradigm with evolutionary computation brings a new quality to the field of metaheuristics, enhancing individuals with possibilities of perception, interaction with other individuals (agents), adaptation of parameters, etc. In the paper such technique—an evolutionary multi-agent system (EMAS)—is compared with a classical evolutionary algorithm (Michalewicz model) implemented with allopatric speciation (island model). Both algorithms are applied to the problem of continuous optimisation in selected benchmark problems. The results are very promising, as agent-based computing turns out to be more effective than classical one, especially in difficult benchmark problems, such as high-dimensional Rastrigin function.

1 Introduction

The paper concerns a hybrid evolutionary-agent [1] approach to solving continuous optimisation problems. In most such applications reported in literature (see e.g., [2] or [3] for a review) an evolutionary algorithm is used by an agent to aid realisation of some of its tasks, often connected with learning or reasoning, or to support coordination of some group (team) activity. In other approaches, agents constitute a management infrastructure for a distributed realisation of an evolutionary algorithm [4]. Yet evolutionary processes are decentralised by nature and indeed one may imagine the incorporation of evolutionary processes into a multi-agent system at a population level [5]. It means that apart from interaction mechanisms typical of MAS (such as communication), agents are able to *reproduce* (generate new agents) and may *die* (be eliminated from the system). A similar idea but with limited autonomy of agents located in fixed positions on some lattice (like in a cellular model of parallel evolutionary algorithms) was developed by e.g., [6]. The key idea of the decentralised model of evolution employed by an *evolutionary multi-agent system* (EMAS) was to ensure full autonomy of agents. Different variants of this model have been successfully applied to different optimisation problems (e.g., single-criteria, multi-criteria, discrete, continuous) [7].

This paper is devoted to the examination of the efficiency of EMAS in the problem of optimisation of a high-dimensional benchmark function. Therefore,

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 132–141, 2013.
© Springer-Verlag Berlin Heidelberg 2013

after recalling the basics of evolutionary, and agent-based computation and presenting the concepts of the examined systems, the experimental setting is given and the results concerning optimisation of 100-dimensional Rastrigin function are discussed for several selected population sizes and allopatric speciation configurations.

2 Evolutionary Agent-Based Optimization

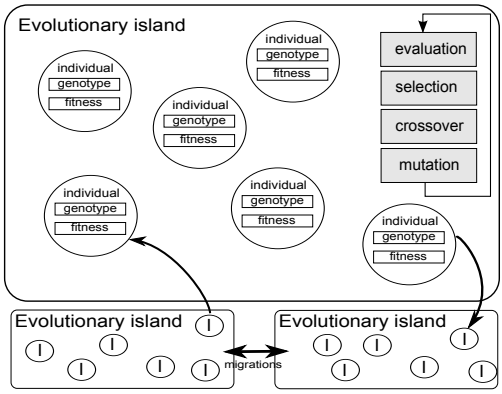
Over the years evolutionary algorithms proved to be an effective universal technique for solving optimisation problems [8]. Instead of directly solving the given problem, subsequent populations of potential solutions are constructed, modelling phenomena of natural evolution. This process consists of two components: selection performed based on the solutions evaluation (fitness function), and generation of new solutions with the use of variation operators (such as crossover and mutation). The process continues until some stopping condition is reached (e.g., number of generations, lack of changes in the best solution found so far).

This kind of search has some drawbacks since many algorithms tend to prematurely loose useful diversity of the population and therefore there is a possibility that the population might get stuck in some part of the search space. To deal with this situation several techniques may be applied, as multi-deme approaches [9]. The schematic presentation of this so-called parallel evolutionary algorithm used as a reference in this paper is presented in Figure 1a. It may be seen that the population of potential solutions is decomposed into evolutionary islands, and there is a possibility of migration between them.

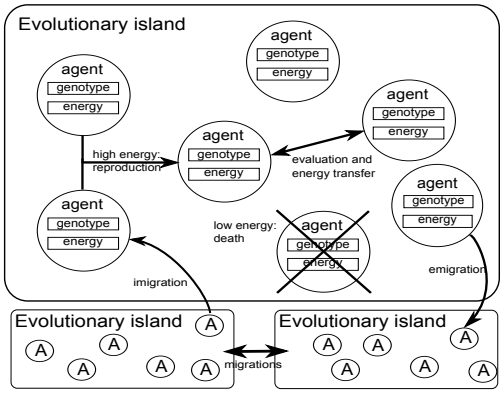
One of the problems with evolutionary algorithms is that the selection of the proper evolutionary algorithm for the given task is an open problem. One of the reasons for that is the weakness of the theory of evolutionary algorithms: "We know that they work, but we do not know why" [8]. What is more, the model of evolution followed by most EAs (with noticeable exceptions) is much simplified and lacks important phenomena observed in organic evolutions, like the one that neither global knowledge nor generational synchronisation is assumed in nature.

The idea of an evolutionary multi-agent system (EMAS) helps to avoid some of the shortcomings of the model of evolution employed in classical evolutionary computation techniques. Agents of EMAS represent solutions for a given optimisation problem like in classical evolutionary algorithms. They may be located on islands, which constitute their local environment where direct interactions may take places, and represent a distributed structure of computation. Obviously, agents are able to change their location, which allows for diffusion of information and resources all over the system.

The main components of evolutionary processes—inheritance and selection—are modelled via agent actions of *death* and *reproduction* (see Fig. 1b). Reproduction consists in the production of a new individual in cooperation with one of its neighbours that is chosen randomly, with the solution inherited from its parent(s) with the use of variation operators (mutation and recombination). Assuming no global knowledge available and the autonomy of agents, selection is based on the non-renewable resources [10].



(a) Parallel evolutionary algorithm (PEA)



(b) Evolutionary multi-agent system (EMAS)

Fig. 1. Schematic presentation of the algorithms discussed in the paper

In the simplest possible model of EMAS there is one type of agents and one resource defined (called *life energy*). Energy is exchanged by agents in the process of evaluation. The agent increases its energy when it finds out that one (e.g., randomly chosen) of its neighbours has lower fitness. In this case, the agent takes part of its neighbour's energy, otherwise, it passes part of its own energy to the evaluated neighbour. The level of life energy triggers agents' actions [5]:

- *Reproduction* – performed when the agent's energy raises above a certain level, a part of energy (usually half of its initial value) is passed to a new agent from each of its parents.
- *Death* – agent is removed from the system when its energy falls below a certain level, the remaining energy is distributed among its neighbours.
- *Migration* – agent (with some probability) may migrate, then it is removed from one evolutionary island and moved to another (random) according to predefined topology.

Each action is attempted randomly with certain probability, and it is performed only when their basic preconditions are met (e.g., an agent may attempt to perform the action of reproduction only if it meets an appropriate neighbour).

The topology of an island, defining the structure of inter-agent relations may be random (full graph of connections between the agents), but in order to enhance diversity of the population, an additional level of population decomposition (beside the evolutionary islands) may be introduced. Thus, a two-dimensional square lattice (similarly to the ones used in Cellular Automata [11]) may be considered. In such lattice, different neighbourhoods (e.g., Moore's) and boundary conditions (e.g., periodic, reflexive and fixed) may be utilised. In such an island, the agents may interact between themselves only providing they are in the zone of each other's neighborhood.

3 Methodology of Experimental Studies

Having vast experience in the development of component-based agent-oriented computing platforms (cf. AgE[1] [12]), a simplified version of a discrete-event simulation and computing system was developed using Python language. The choice of this technology was conditioned by a relatively easy implementation process and high portability [13]. Using this software environment, both a classical evolutionary algorithm (as proposed by Michalewicz [14]) and EMAS were similarly implemented and configured. Another important assumption of the experimental study was that all possible parameters of both systems were set to the same values. For each considered case the experiments were repeated 30 times and the standard deviation (or other statistical measures, such as median and appropriate quartiles for box-and-whiskers plots) was computed as a measure of repeatability. The results presented below concern the observation of the best fitness according to the step of computation, moreover, the observation of diversity was also performed using two selected definitions of this measure:

MOI – Morrison-De Jong measure based on concept of moment of inertia for measurement of mass distribution into arbitrarily high dimensionality spaces [15],

MSD – maximum standard deviation of each gene computed for all individuals in the population.

Common configuration of the algorithms

- Representation: real-valued.
- Mutation: normal distribution-based modification of one randomly chosen gene.
- Crossover: single-point.
- Migration topology: 3 fully connected islands.
- Migration probability: 0.01 per agent/individual (each one migrates independently—possibly to different islands).

[1] `http://age.iisg.agh.edu.pl`

EMAS configuration

- Initial energy: 100 units received by the agents in the beginning of their lives.
- Evaluation energy win/loose: 20 units passed from the looser to the winner.
- Minimal reproduction energy: 90 units required to reproduce.
- Death energy level: 0, such agents should be removed from the system.
- Boundary condition for the intra-island lattice: fixed, the agents cannot cross the borders.
- Intra-island neighbourhood: Moore's, each agent's neighbourhood consists of 8 surrounding cells.
- Size of 2-dimensional lattice as an environment: 10x10.

Configuration for preliminary experiments

- Stop condition: 3000 steps of experiment.
- Benchmark problems: Ackley, DeJong, Rastrigin, and Rosenbrock functions [16].
- Problem size: 50 dimensions.
- For EMAS the evaluation energy win/loose was set to 40 units.
- Population size: 30 individuals/agents located on each island.

Configuration for main experiments

- Stop condition: 100000 steps of experiment.
- Benchmark problem: Rastrigin function [16].
- Problem size: 100 dimensions.
- Population size configurations:
  - 25 individuals on 1 island,
  - 25 individuals on 3 islands,
  - 40 individuals on 1 island,
  - 40 individuals on 3 islands.

4 Experimental Results

In the preliminary experiments, selected benchmark functions were tackled [16] with both EMAS and PEA (Michalewicz version [14]). The results point out, that EMAS turns out to be much more effective in localising optimum of the problems tackled for all tested benchmark functions 2. Though PEA converges faster, it seems to be stuck in a local extremum of the function, while EMAS overcomes this problem and produces significantly better solutions.

In main experiments, different configurations for higher dimensionality (100 dimensions) were tested. In Figure 3, a simple comparison of the results of computation, obtained for one-population configuration (1 islands, 40 individuals), namely observation of the best fitness in each step of the computation may be seen. It turns out that EMAS outperforms PEA for over two orders of magnitude. This is a very promising result, and it may be further verified by checking

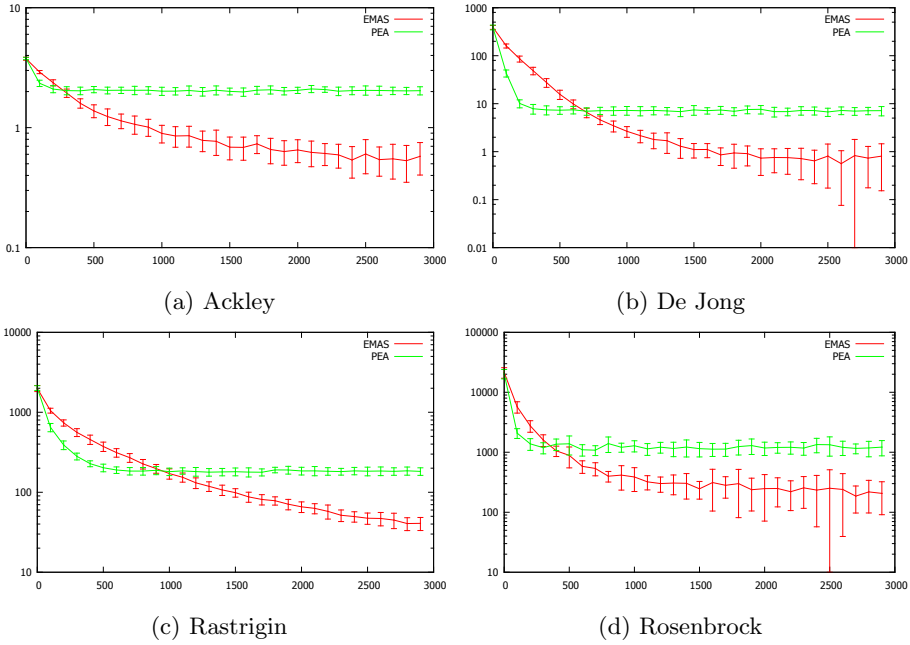


Fig. 2. Comparing of PEA and EMAS best fitness for selected 50-dimensional benchmark problems

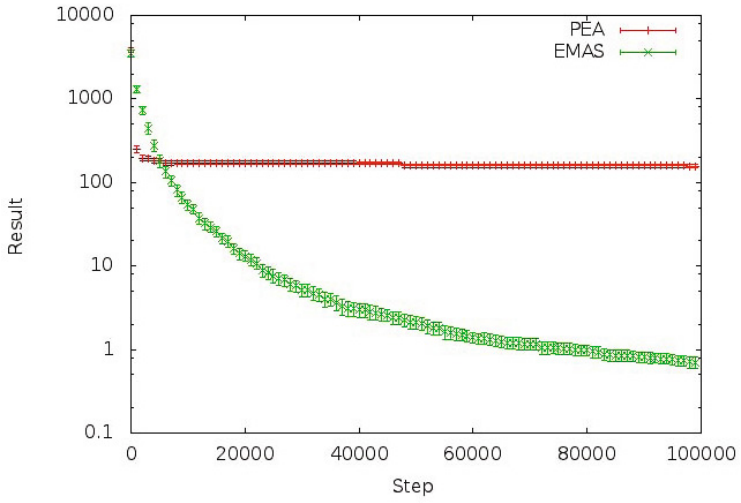
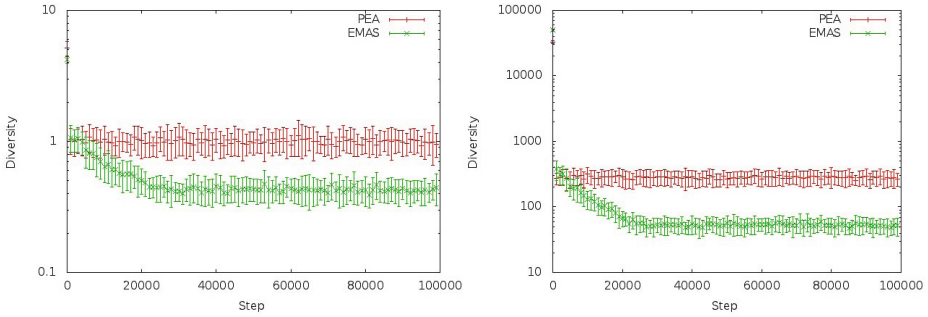


Fig. 3. Result comparison—EMAS vs PEA (1 island, 40 agents/individuals) with standard deviations

Table 1. EMAS and PEA optimization results obtained for 25 and 40 individuals

	1 island		3 islands	
	EMAS	PEA	EMAS	PEA
25 individuals				
Result	1.77	242.96	0.71	156.34
St. Dev.	0.22	6.54	0.10	6.84
St. Dev. %	12.24	2.69	14.46	4.38
40 individuals				
Result	0.90	180.48	0.37	111.45
St. Dev.	0.13	6.45	0.05	4.16
St. Dev. %	15.24	3.57	13.56	3.73



(a) MSD diversity (b) MOI diversity

Fig. 4. Diversity computed according to MOI and MSD schemes, 1 island 40 agents/individuals with standard deviations

the results obtained in both computations in the last step that are presented in Table 1.

In this table, the results obtained for other configurations (3 islands, 40 individuals) were also presented. In these cases the domination of EMAS is still retained. It is easy to see that the results are repeatable (as standard deviation is relatively low).

Another important information is that the multi-population models of computation tends to be better than single-population, as these former have better capabilities of exploration (as the population is decomposed), still retaining the capability of exploitation (in each single sub-population). It is to note also that increasing the number of individuals (from 25 to 40) improved the final result (although this requires further proving and many more experimental cases).

In Figures 4a and 4b, the diversity computed according to MSD and MOI schemes were shown. It is easy to see that EMAS has lower diversity than PEA in both cases. However, it turns out that it does not hampers the computing efficiency (as EMAS outperforms PEA). Moreover, the diversity, though lower,

Table 2. EMAS and PEA MSD and MOI diversity for 25 and 40 individuals

	1 island		3 islands	
	EMAS	PEA	EMAS	PEA
25 individuals				
Result	0.56	0.99	0.65	1.31
St. Dev.	0.22	0.14	0.29	0.14
St. Dev. %	38.67	14.04	44.05	10.87
40 individuals				
Result	0.50	1.00	0.57	1.30
St. Dev.	0.23	0.14	0.30	0.15
St. Dev. %	45.01	13.84	52.35	11.16

(a) MSD diversity

	1 island		3 islands	
	EMAS	PEA	EMAS	PEA
25 individuals				
Result	77.51	287.63	271.64	1253.43
St. Dev.	903.60	1004.92	2883.95	3000.84
St. Dev. %	1165.86	349.37	1061.69	239.41
40 individuals				
Result	135.22	313.53	460.94	1340.27
St. Dev.	1590.73	1030.57	4962.78	3066.69
St. Dev. %	1176.37	328.69	1076.66	228.81

(b) MOI diversity

Table 3. EMAS and PEA average and maximum number of steps between subsequent improvements of the computation result for 25 and 40 individuals

	1 island		3 islands	
	EMAS	PEA	EMAS	PEA
25 individuals				
Result	116.90	133.74	142.61	195.33
St. Dev.	4.26	72.37	5.12	72.90
St. Dev. %	3.64	54.12	3.59	37.32
40 individuals				
Result	119.07	166.13	147.16	234.57
St. Dev.	4.43	87.15	5.97	77.02
St. Dev. %	3.72	52.46	4.05	32.84

(a) Average number of steps

	1 island		3 islands	
	EMAS	PEA	EMAS	PEA
25 individuals				
Result	4042.37	27441.57	3940.23	42225.17
St. Dev.	1580.74	17264.97	1330.81	17906.04
St. Dev. %	39.10	62.92	33.77	42.41
40 individuals				
Result	3850.93	33019.87	3596.30	47966.66
St. Dev.	1173.24	23218.82	889.32	20118.41
St. Dev. %	30.47	70.32	24.73	41.94

(b) Maximum number of steps

is still quite stable (see standard deviation range marked on the graphs) that leads to the conclusion that though the population in EMAS is not so diverse as in PEA, the exploration and exploitation features of the system are balanced.

The final values of both diversity measures (MOI and MSD) shown in Tables 2a, 2b confirm the observation of the behaviour of the computation observed in Figures 4a and 4b.

In order to examine the dynamics of the computing process for PEA and EMAS, average number of steps between subsequent improvements of the best fitness observed are presented in Table 3a. It is easy to see that also in this case, EMAS outperforms PEA. This result confirms that lower diversity of EMAS in comparison to PEA does not hamper the capability of improving the result of the computation.

Another confirmation of the above observation may be found when looking at Table 3b. There, maximum number of steps, required for improving the value of the fitness function were shown. It is easy to see that EMAS outperforms PEA also in this case.

5 Conclusions

Search for complex computing paradigms will be always justified, at least as long as new problems will be found (cf. "no free lunch theorem" by Wolpert and Macready [17]). On the other hand, one must remember about famous "Occham razor" principle, and avoid creation of unnecessary, complex computing systems, just for the sole creation.

High dimensional, multi modal problems require dedicated systems, appropriately suited to their features. Biologically-inspired computing systems seem to be a good answer to such requirements, connected with solving such complex problems, as maintaining the balance between exploitation and exploration, by stabilizing its diversity, still being able to escape from the danger of premature convergence.

In this paper, the evolutionary multi agent-system was recalled and compared to a classical evolutionary algorithm. It turns out that conceptually, the agent-oriented approach differs mainly in the model of selection and ontogenesis. Although as one of the results, lower diversity was observed in EMAS than in PEA, it was still maintained in stable level. Other experiments proven that the dynamics of EMAS turned out to be better than PEA, in the means of approaching the global minimum (as average and maximum steps required for improving the fitness function were tested). However, the main result concernes the attained solution. Namely in the case of EMAS, for high dimensional Rastrigin benchmark function, is over two orders of magnitude better than the one observed for PEA. This makes EMAS a promising tool to solve complex problems.

Predictably, the multi-population systems were better in the means of measured parameters (best fitness, average number of steps between improvement of the results, etc.), this feature was observed both for PEA and EMAS. Other predictable result, regarding increasing the number of individuals in the populations, affected positively the quality of the final solution in both cases (EMAS and PEA), however these results may be treated as preliminary, as only two experimental cases were tested.

In the future, further testing is planned, including covering a broader range of benchmark functions, as well as real-world problems. Other modifications of classical and evolutionary-agent computing models (such as immunological or memetic) will also be examined.

Acknowledgment. The research presented here was partially supported by the grant "Biologically inspired mechanisms in planning and management of dynamic environments" funded by the Polish National Science Centre, No. N N516 500039.

References

1. Kisiel-Dorohinicki, M., Dobrowolski, G., Nawarecki, E.: Agent populations as computational intelligence. In: Rutkowski, L., Kacprzyk, J. (eds.) Neural Networks and Soft Computing. Advances in Soft Computing, Physica-Verlag (2003)

2. Sarker, R., Ray, T.: Agent-Based Evolutionary Search. Springer (2010)
3. Chen, S.H., Kambayashi, Y., Sato, H.: Multi-Agent Applications with Evolutionary Computation and Biologically Inspired Technologies. IGI Global (2011)
4. Schaefer, R., Kołodziej, J.: Genetic search reinforced by the population hierarchy. Foundations of Genetic Algorithms 7 (2003)
5. Kisiel-Dorohinicki, M.: Agent-Oriented Model of Simulated Evolution. In: Grosky, W.I., Plášil, F. (eds.) SOFSEM 2002. LNCS, vol. 2540, pp. 253–261. Springer, Heidelberg (2002)
6. Zhong, W., Liu, J., Xue, M., Jiao, L.: A multiagent genetic algorithm for global numerical optimization. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 34(2), 1128–1141 (2004)
7. Byrski, A., Dreżewski, R., Siwik, L., Kisiel-Dorohinicki, M.: Evolutionary multi-agent systems. The Knowledge Engineering Review (2012) (accepted for publication)
8. Back, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: Comments on the history and current state. IEEE Trans. on Evolutionary Computation 1(1) (1997)
9. Cantú-Paz, E.: A summary of research on parallel genetic algorithms. IlliGAL Report No. 95007. University of Illinois (1995)
10. Dreżewski, R., Cetnarowicz, K.: Sexual Selection Mechanism for Agent-Based Evolutionary Computation. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007, Part II. LNCS, vol. 4488, pp. 920–927. Springer, Heidelberg (2007)
11. Wolfram, S.: A New Kind of Science. Wolfram Media (2002)
12. Byrski, A., Kisiel-Dorohinicki, M.: Agent-Based Model and Computing Environment Facilitating the Development of Distributed Computational Intelligence Systems. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part II. LNCS, vol. 5545, pp. 865–874. Springer, Heidelberg (2009)
13. Lutz, M.: Programming Python. O'Reilly Media (2011)
14. Michalewicz, Z.: Genetic Algorithms Plus Data Structures Equals Evolution Programs. Springer-Verlag New York, Inc., Secaucus (1994)
15. Morrison, R.W., De Jong, K.A.: Measurement of Population Diversity. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 31–41. Springer, Heidelberg (2002)
16. Digalakis, J., Margaritis, K.: An experimental study of benchmarking functions for evolutionary algorithms. Int. J. of Computer Mathematics 79(4), 403–416 (2002)
17. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)

Domestic Load Scheduling Using Genetic Algorithms

Ana Soares[1], Álvaro Gomes[1,2], Carlos Henggeler Antunes[1,2], and Hugo Cardoso[2]

[1] INESC Coimbra, Rua Antero de Quental 199, 3000-033 Coimbra, Portugal
argsoares@gmail.com
[2] Dept. of Electrical Engineering and Computers, University of Coimbra, Coimbra, Portugal
{agomes,ch}@deec.uc.pt, hugodcardoso@gmail.com

Abstract. An approach using a genetic algorithm to optimize the scheduling of domestic electric loads, according to technical and user-defined constraints and input signals, is presented and illustrative results are shown. The aim is minimizing the end-user's electricity bill according to his/her preferences, while accounting for the quality of the energy services provided. The constraints include the contracted power level, end-users' preferences concerning the admissible and/or preferable time periods for operation of each load, and the amount of available usable power in each period of time to account for variations in the (non-manageable) base load. The load scheduling is done for the next 36 hours assuming that a dynamic pricing structure is known in advance. The results obtained present a noticeable decrease of the electricity bill when compared to a reference case in which there is no automated scheduling.

Keywords: Domestic Load Scheduling, Genetic Algorithms, Electric Loads, Automated Energy Management.

1 Introduction

The evolution of power systems towards smart grids will expectedly foster the implementation of dynamic tariffs requiring a more proactive attitude from the typical domestic end-user to optimize electricity use in face of dynamic variables such as electricity prices, weather conditions, comfort requirements, and local generation availability. However, this is a very challenging task and the effort required to reach an optimal solution is excessive for most users, namely for the elderly or those working outside home, blocking them from effectively achieving benefits from those tariff structures. In that context, some form of automated decision support system to achieve optimal solutions is needed. Therefore, adequate algorithms able to manage different loads without depreciating the quality of the energy services provided must be developed.

The main objective of scheduling domestic loads is helping users to take advantage of different alternatives of using energy services and reducing the electricity bill. This can be done by implementing some demand response actions: shifting appliances to time periods when the kWh price is lower; interrupting the working cycles for short periods of time and/or changing the parameterization of some loads, namely temperature parameters in heating, ventilation and cooling systems (HVAC) [1–4, 13–15].

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 142–151, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Considering a smart grid scenario and the existence of dynamic tariffs, some studies focus on domestic load control. This control mainly uses scheduling optimization algorithms to allocate loads in a certain time period [1, 2, 5–9, 11–13]. The study presented in [1] depicts an approach to calculate the optimal starting point for different appliances while assuring that the load limitation curve is not exceeded. The electrical load management problem is modeled as a nonlinear integer programming problem and the methodology used to minimize energy costs to the consumer and the violation of the load limitation curve is a customized evolutionary algorithm with local search. This method is used to implement demand response actions over electric vehicles, washing machines, dryers and dishwashers. In [11], an optimization framework is developed to achieve a desired trade-off between the minimization of the electricity bill and the waiting time for the operation of each appliance. The methodology chosen considers almost real-time prices and price forecasts to calculate the optimal scheduling operation and energy consumption for each appliance. Another methodology designed to optimize the use of several energy services from the end-users' perspective is presented in [13], which uses Particle Swarm Optimization to maximize the net benefits resulting from the total energy service benefits minus the costs of energy provision.

The main difference between our work and those mentioned above is the incorporation of end-users' preferences regarding admissible and preferable time slots for load operation. Moreover, it considers both load demand maximum value, which should be as far as possible from contracted power values, and the amount of available usable energy in each period of time, which should be as high as possible.

This section provided the interest and motivation for the study. Section 2 focuses on the description of the problem while in Section 3 the case study and the simulation results are presented. Conclusions and future work are outlined in Section 4.

2 Problem Description

The aim of this study is to develop a model and an algorithm to minimize the electricity bill for a domestic end-user taking into account his/her preferences, electricity prices and available contracted power. This is achieved by implementing control actions over manageable loads, namely shiftable loads. Shiftable loads can have the working cycle delayed or anticipated while respecting end-users' preferences [16]. Loads like cloth washing machines, cloth dryers, dishwashers, electric vehicles and electric water heaters may be included in this category. It is still considered that a base load exists that is not available to be changed by the scheduling algorithm, such as ovens, entertainment systems, lighting, etc. This assumption is made since the end-user is not willing to give up the energy services provided by those loads.

The complexity of this problem arises from the combination of the allocation of shiftable loads over time, the kWh price variation, the existence of more than one level for the contracted power, the need to assure the continuity of the energy services provided and the end-users' preferences concerning the time slots. Time slots are the periods of the day in which the end-user indicates, by assigning different degrees of preference, the energy service should be provided. When the same energy service can

be provided in different time slots, each one of them may be characterized by different preference levels. Time slot preferences should be respected as closely as possible to reduce the discomfort caused to the end-users and thus maintain a high rate of acceptance of the proposed schedule.

A genetic algorithm (GA) is used to perform the search of the solution space coping with the combinatorial nature of the problem. Genetic algorithms are based on the mechanisms of natural genetics and selection and are used to perform an efficient and effective search to find the global optimal solution to a single-objective optimization problem or make the characterization of the non-dominated solution set to a multi-objective optimization problem. The solutions (individuals) population is dealt with three main operators to produce new offspring for the next generation. These operators are selection, crossover, and mutation. In each generation the population is evaluated to compute the fitness of every member and a group of individuals is selected based on their performance. Therefore, some of these individuals may undergo reproduction by means of the crossover operator and possibly mutate to give rise to a new generation (potential solutions) [10]. The process repeats until a convergence criterion is satisfied.

In our problem, a solution is encoded by a string of integers, each position denoting the minute in which each load begins its working cycle. Solutions are therefore represented by a vector of integers $x = (x_1, x_2, \dots, x_{N_c})$ where $x_j \in \{0, \dots, N_t\}$, with $N_t = 2160$, is the minute in which load $j \in \{1, \dots, N_c\}$ starts its working cycle. N_c is the number of loads to be scheduled. The working cycle to be scheduled within the planning period is known for each load (see section 3).

A crossover operator based on a random mask was implemented combining the features of the two parents, selected through a ternary tournament, to form two new children by swapping their chromosome segments. The crossover operator is responsible for the information exchange between potential solutions by switching the initial instant of one or more operation cycles between different loads (Fig. 1). The mutation operator may introduce some extra variability into the population by implementing small displacements in the starting minute of the working cycles of loads. In this case, the mutation operator changes the initial instant of a given load (gene) within a given range [-δ,+δ], in which δ is a deviation bound, in minutes. (Fig. 2).

Load	1	2	3	4
Crossover?	Yes	No	Yes	Yes
Parent 1	1200	30	550	400
Parent 2	1189	49	589	369
Child 1	1189	30	589	369
Child 2	1200	49	550	400

Fig. 1. Example of crossover considering 4 loads

Loads	1	2	3	4
Child 1*	1189	25	589	369
Child 2*	1200	49	555	400

Fig. 2. Example of mutation of the two children in Fig. 1

The objective function is the overall cost to be minimized, which is the sum of the following terms: cost of the energy consumed by the loads being managed, (monetized) cost of the penalties associated with the end-users' preferences concerning the time slots for the operation of each load, (monetized) cost of the penalty associated with the closeness of the actual peak power with respect to the contracted power (as a proxy for the possibility of interruption of the energy services provided if there is an unexpected variation of the base load, which is not being controlled), and the (monetized) cost associated with the available energy in each time slot (the penalty increases if the energy available decreases).

The penalties associated with the end-users preferences are incorporated into the objective function since there is a cost inherent to the time slots: the cost is zero if the load is operating in the preferred time slot, the cost may be extremely high in those periods of time where the end-user does not allow a certain working cycle to be allocated (thus turning this preference into a hard constraint). Moderate, increasing or decreasing costs may also be assigned to the time slots, thus turning these preferences into soft constraints, as it will be presented in section 3.

In order to better deal with possible unexpected variations of the base load, load schedules presenting higher peak power values and lower available energy in the time slots are penalized. These penalties are also monetized and encompassed in the objective function in order to keep the power peaks as far as possible from the contracted power and the energy available in each time slot as high as possible (lower demand patterns) (Fig. 3 and Fig. 4). Therefore, for each instant in the time slot where the working cycle may be allocated, the penalty will be higher when the difference between the contracted power and the power peak (ΔP) is smaller. Additionally, the available energy is also considered in the objective function: if the amount of energy still available to be consumed in the time slot is high, taken into account the contracted power level, then the penalty is small. If the amount of energy available for time slot decreases, the penalty will increase.

In this approach the contracted power is considered as a hard constraint and thus cannot be violated to prevent the occurrence of interruption of the energy services.

Since the performance of genetic algorithms is highly dependent on the parameters chosen, the identification and selection of a suitable value or range of values for every parameter is a crucial step. Those values were tuned through experimentation and are presented in Table 1. The value considered for the mutation operator is high in order to guarantee the construction of good diversified solutions [10]. An elitist strategy was considered to preserve in the population a number of individuals with the best fitness. The elite set is composed by 10 individuals. These individuals replace new ones generated with the worst fitness.

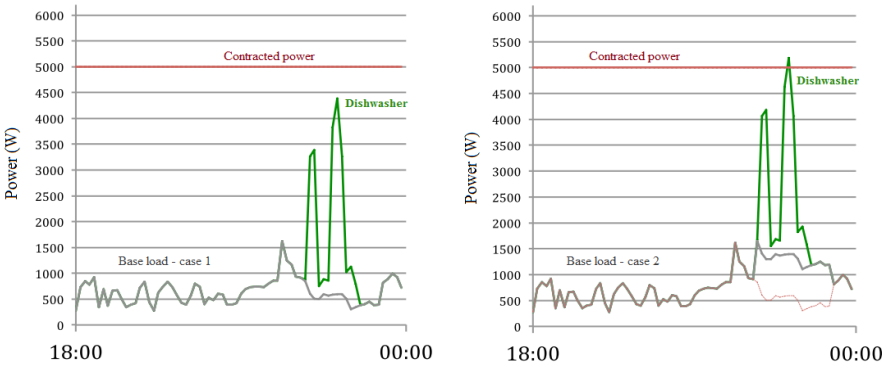


Fig. 3. Base load variation and its implication on the total power required, which would lead to the interruption of the energy services provided (an infeasible solution in our approach)

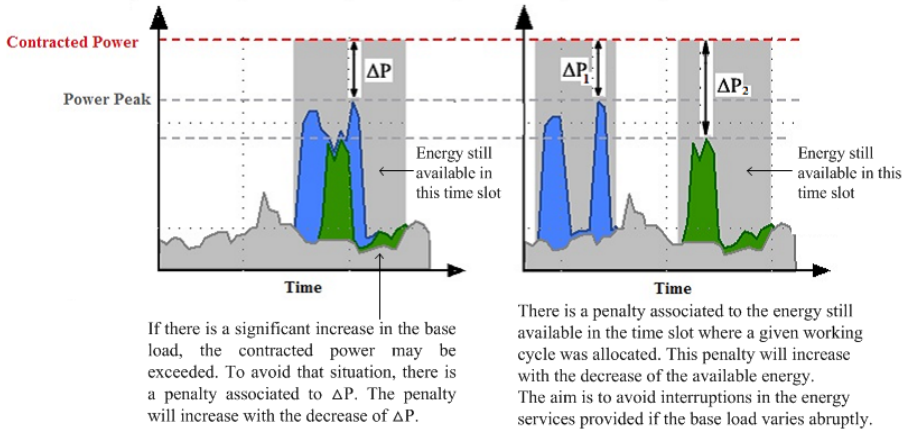


Fig. 4. Constraints included in the model to prevent the interruption of the energy services provided when facing unexpected base load variation

Table 1. Some characteristics of the genetic algorithm

Parameters		
General	Size of population	200
	Number of generations	100
Selection	Number of participants in the tournament	3
Crossover	Crossover Rate	0.5
Mutation	Mutation rate	0.8
	Deviation bound (min.)	5

3 Case Study and Simulation Results

The case study considers an individual house and the loads to be scheduled are cloth washing machines (CWM – load 1 and load 3), dishwasher (DW – load 2), cloth dryer (CD –load 4), electric water heater (EWH – load 5) and electric vehicle (EV – load 6) (Fig. 5). For these five loads, the type of control applied consists in shifting the working cycles over time. In the case of the electric water heater, due to the dissociation that may exist between the energy service provided and the respective electricity consumption, the possible increase of heat losses associated with the displacement of the working cycle was considered by increasing the period during which the EWH is on. It was considered that the EV can be charged at power levels multiple of 1 kW up to 6 kW. In this case study the amount of energy to be provided to the EV is 20 kWh.

Real working cycles and a small time step - 1 minute - have been considered to make this model as close as possible to reality. This time resolution ensures that any power peaks that could be too close to the contracted power or even overcome it causing the interruption of the energy services provided are taken into account.

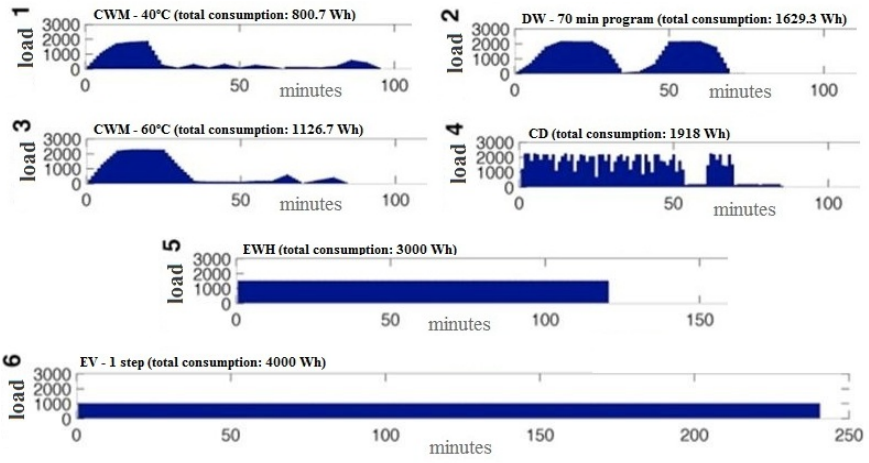


Fig. 5. Working cycles of the manageable loads

The contracted power is considered as a hard constraint and has two levels: 3.45 kW between 4 am and 8 pm and 6.9 kW between 8 pm and 4 am (Fig. 6). The tariff structure considered is also presented in Fig. 6. The end-user's preferences concerning the time slots for each manageable load are displayed in Fig. 7. These preferences can be seen as hard or soft constraints depending on the situation. For example, the end-user identifies two possible time slots for the operation of load 1 (CWM), even though with different preferences associated with them. In this case and for this load, the end-user

prefers the first time slot (no penalty is associated with it), but he/she is also willing to accept the functioning of the CWM in the second time slot (but in this case with a high penalty stating that he/she accepts the operation in this time slot but he/she wants to disfavor it). Therefore this is a soft constraint modeling the higher or lower willingness of the user to accept the working cycle to be scheduled in specified time slots. The end-user may also specify the periods in which the loads are not allowed to operate, which may happen due to several reasons, and therefore this is modeled as a hard constraint. Several other preference structures and levels are also shown in Fig. 7 including time increasing or decreasing preferences to schedule load operation. The penalty coefficients, which may be adjusted to take into account different contexts and/or user profiles, are monetized to be included in the overall cost objective function.

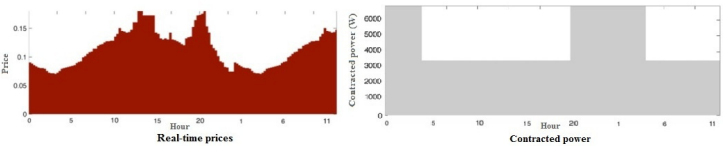


Fig. 6. Tariff structure and contracted power

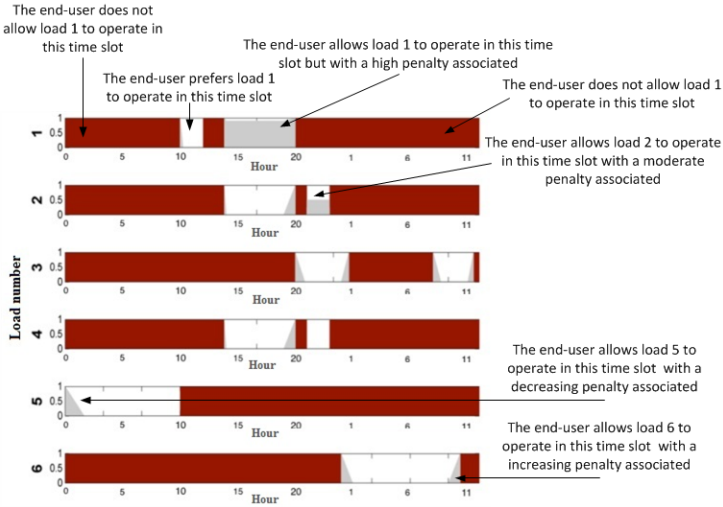


Fig. 7. End-user time slot preferences for each load

The problem of load scheduling consists in allocating the cycles of each load in the most adequate time slot, minimizing the overall cost while respecting the constraints associated with each appliance. Since the load scheduling is done based on demand forecasts for the next day and a half (36 hours = 2160 minutes), it is not possible to

accurately predict the uncontrollable demand. Thus, in order to avoid exceeding the contracted power, solutions presenting lower demand (higher available power) in the periods of time in which loads were allocated are preferred. The use of the penalty allows benefiting such solutions, thus contributing to reduce the possibility of occurrence of any interruption of the energy services when unexpected variations in the base load happen (Fig. 3 and Fig. 4).

When analyzing the solutions obtained for this load scheduling problem for 30 runs, the minimum cost achieved is 2.521 monetary units (m.u.) while the worst minimum value is 2.523 m.u.. The average of the best solutions for the 30 runs is 2.522 m.u.. The best solutions for the 30 runs are displayed in Fig. 8 converging to an optimal solution. For the preferences and constraints presented in Fig. 6 and Fig. 7 it can be seen that the algorithm converges well before reaching 100 generations. The scheduling obtained for the best solution computed is displayed in Fig. 9.

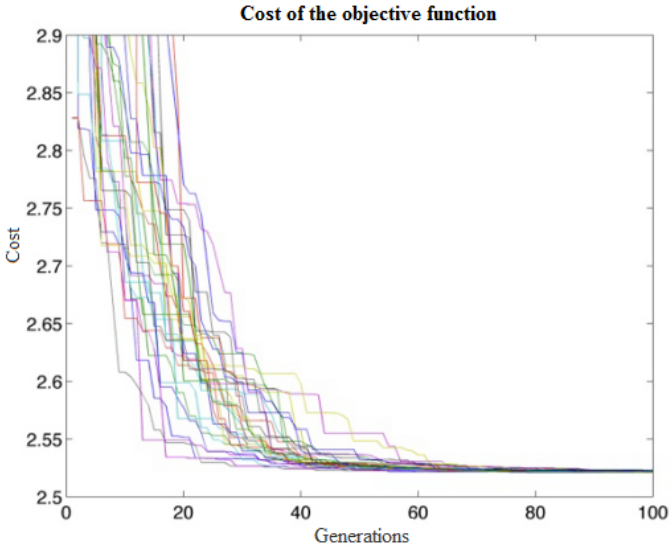


Fig. 8. Best cost achieved for the objective function in each generation of the 30 runs

Up to 40% of savings have been achieved when comparing the results obtained by using the proposed approach with a reference case. In the reference case, loads are operated in the end-users' preferred time slot (not penalized time slots in Fig. 6), only guaranteeing that the contracted power is not exceeded. In that situation the average minimum cost was 4.218 m.u.. The savings achieved may still increase in other tariff structures, especially if there is a higher variation between the kWh prices along the day.

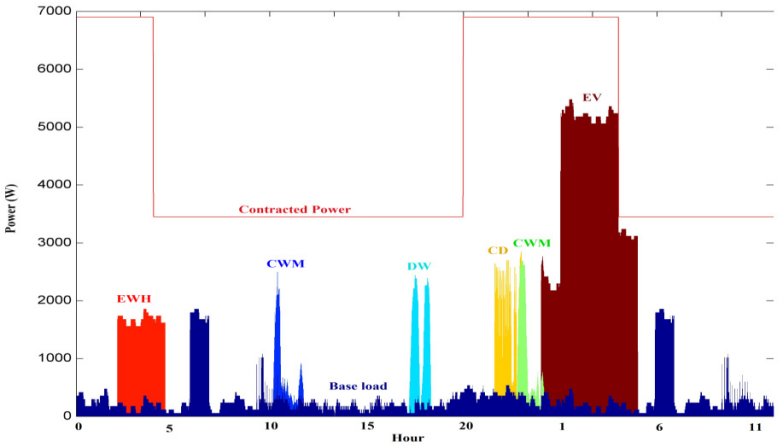


Fig. 9. Best solution

4 Conclusion and Future Work

An approach based on GA has been presented, which is aimed at minimizing the electricity bill by scheduling end-use loads while assuming a real-time pricing structure and considering several end-user' preferences regarding the energy service provided by loads. Additionally, the use of penalties to make less attractive solutions presenting lower levels of available energy (higher demand levels) in the time slots together with the inclusion of end-users' preferences attempts to guarantee end-users' satisfaction. Savings for a real case study were computed and compared to a reference case where there is no automated scheduling.

These algorithms may be implemented in energy management systems, such as the one proposed in [8], to coordinate the different energy resources within a house. To simplify the process of inserting information the system must incorporate an intuitive interface so that the end-user may easily select which loads should be scheduled and under which time slots preferences. For this purpose, a-priori defined profiles may be available to facilitate the users' tasks.

Future work will include other energy resources (for instance, local microgeneration and stationary storage systems), more automated demand response actions (e.g., temperature reparameterization and short-period interruptions for other loads) and different tariff schemes. The genetic algorithm may be improved regarding adaptive parameters and dynamic environments. Models taking explicitly into consideration the multiple objective characteristic of the problem are being developed. These models include the minimization of energy and power costs and the maximization of the quality of the energy services provided, namely end-user's comfort as captured by his/her preferences expression.

Acknowledgements. This work has been framed under the Energy for Sustainability Initiative of the University of Coimbra and supported by Energy and Mobility for Sustainable Regions Project CENTRO-07-0224-FEDER-002004 and Fundação para a Ciência e a Tecnologia (FCT) under grant SFRH/BD/88127/2012, and project grants MIT/SET/0018/2009 and PEst-C/EEI/UI0308/2011.

References

1. Allerding, F., Premm, M., Shukla, P.K., Schmeck, H.: Electrical Load Management in Smart Homes Using Evolutionary Algorithms. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 99–110. Springer, Heidelberg (2012)
2. Du, P., Lu, N.: Appliance Commitment for Household Load Scheduling. IEEE Transactions on Smart Grid 2(2), 411–419 (2011)
3. Gomes, A., et al.: A Multiple Objective Evolutionary Approach for the Design and Selection of Load Control Strategies. IEEE Transactions on Power Systems 19(2), 1173–1180 (2004)
4. Gudi, N., et al.: Demand response simulation implementing heuristic optimization for home energy management. In: North American Power Symposium 2010, pp. 1–6. IEEE (2010)
5. He, Y., et al.: Optimal Scheduling for Charging and Discharging of Electric Vehicles. IEEE Transactions on Smart Grid 3(3), 1095–1105 (2012)
6. Koutitas, G.: Control of Flexible Smart Devices in the Smart Grid. IEEE Transactions on Smart Grid 3(3), 1333–1343 (2012)
7. Kwag, H.-G., Kim, J.-O.: Optimal combined scheduling of generation and demand response with demand resource constraints. Applied Energy 96, 161–170 (2012)
8. Livengood, D., Larson, R.: The Energy Box: Locally Automated Optimal Control of Residential Electricity Usage. Service Science 1(1), 1–16 (2009)
9. Lu, N.: An Evaluation of the HVAC Load Potential for Providing Load Balancing Service. IEEE Transactions on Smart Grid 3(3), 1263–1270 (2012)
10. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer (1992)
11. Mohsenian-Rad, A.-H., Leon-Garcia, A.: Optimal Residential Load Control With Price Prediction in Real-Time Electricity Pricing Environments. IEEE Transactions on Smart Grid 1(2), 120–133 (2010)
12. Molina, A., et al.: Implementation and assessment of physically based electrical load models: application to direct load control residential programmes. IEE Proceedings - Generation, Transmission and Distribution 150(1), 61 (2003)
13. Pedrasa, M.A.A., et al.: Coordinated Scheduling of Residential Distributed Energy Resources to Optimize Smart Home Energy Services. IEEE Transactions on Smart Grid 1(2), 134–143 (2010)
14. Pedrasa, M.A.A., et al.: Scheduling of Demand Side Resources Using Binary Particle Swarm Optimization. IEEE Transactions on Power Systems 24(3), 1173–1181 (2009)
15. Roe, C., et al.: Simulated demand response of a residential energy management system. In: IEEE 2011 EnergyTech, pp. 1–6. IEEE (2011)
16. Soares, A., et al.: Domestic load characterization for demand-responsive energy management systems. In: 2012 IEEE International Symposium on Sustainable Systems and Technology (ISSST), pp. 1–6. IEEE, Boston (2012)

Evolutionary Algorithm Based Control Policies for Flexible Optimal Power Flow over Time

Stephan Hutterer[1], Michael Affenzeller[1,2], and Franz Auinger[1]

[1] Upper Austria University of Applied Sciences
{stephan.hutterer,franz.auinger}@fh-wels.at
[2] Josef Ressel Center Heureka!
michael.affenzeller@heuristiclab.com

Abstract. General optimal power flow (OPF) is an important problem in the operation of electric power grids. Solution methods to the OPF have been studied extensively that mainly solve steady-state situations, ignoring uncertainties of state variables as well as their near-future. Thus, in a dynamic and uncertain power system, where the demand as well as the supply-side show volatile behavior, optimization methods are needed that provide solutions very quickly, eliminating issues on convergence speed or robustness of the optimization. This paper introduces a policy-based approach where optimal control policies are learned offline for a given power grid based on evolutionary computation, that later provide quick and accurate control actions in volatile situations. With such an approach, it's no more necessary to solve the OPF in each new situation by applying a certain optimization procedure, but the policies provide (near-) optimal actions very quickly, satisfying all constraints in a reliable and robust way. Thus, a method is available for flexible and optimized power grid operation over time. This will be essential for meeting the claims for the future of smart grids.

1 Introduction

Optimization plays an essential role in power grid operation, offering a wide range of different practical applications for example in power flow studies, maintenance scheduling or infrastructure planning. Here, numerous findings in system engineering and analysis as well as actual achievements in computer sciences enabled an established usage of optimization in todays operation centers. Starting its application in the early 1970s, the general class of OPF aims at finding an optimal configuration of controllable units (generator real power output, transformer tap setting, FACTS devices, etc.) within a given power system for minimizing some cost function with respect to operational constraints, where different extensions exist in both literature and practice [15] [9]. Based on these essential formulations, further optimization applications have been identified within the last decades such as infrastructure planning issues or computation of optimal maintenance schedules of supply, transmission or distribution equipment [9], that also mostly base on the general optimal power flow formulation.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 152–161, 2013.
© Springer-Verlag Berlin Heidelberg 2013

However, all general OPF-approaches principally aim at finding a solution for a steady-state situation (defined by the load and supply configuration at a specific time step), ignoring uncertainties of various state variables as well as their near-future. Thus, in a dynamic and uncertain power system, where the demand as well as the supply-side show volatile behavior, optimization methods are needed that provide solutions very quickly, eliminating problems with convergence speed or robustness of the optimization. Therefore, a policy-based approach is introduced within this paper, where optimal control-policies are learned offline for a given power grid based on evolutionary computation, that later provide quick and accurate control actions in volatile situations. The validity of this approach will be demonstrated by applying it to the well known benchmark problem; the IEEE 30-bus test system. Comparisons are made between the proposed approach and the exact OPF solution by comparing the achievable qualities of the flexible policie's actions with the exact OPF solution in examplarily fixed situations that build the test set for analysis.

The rest of the paper is organized as follows: the next section introduces the general aim and a literature review on non-stationary optimization over time. Section 3 proposes the evolution of flexible policies that are built upon so called "atomic rules". Since the formulation of those "atomic rules" is crucial to the success of the approach, they will be discussed in section 4. Section 5 then describes the experimental evaluation yielding to analysis that show the approach's validity. Finally, section 6 rounds up the paper with some concluding remarks as well as an outlook to further investigations.

2 Non-stationary Optimization

Many task in the optimization of power grids are naturally dynamic and non-stationary. Consider the general OPF as stated earlier, the aim is to find the optimal configuration of all controllable units for satisfying a given load situation, using steady-state representation of the power grid. Thus, the solution of this problem addresses exactly one stationary state $J(t)$, disregarding possible states in the near future or eventual uncertain conditions in the system. Considering the system one time step later ($J(t+1)$) due to changing conditions of weather, customer-behavior or any other influence, the power flow in the system would change, hence, requiring a new solution to the optimal power flow problem further necessitated by the non-linear behavior of an electric power distribution system. Such a new computation would require a robust and fast-converging solution method, that guarantees quick support with a new optimal solution, independent of system complexity and starting point. This cannot be guaranteed by traditional OPF methods [12]. Thus, electric power systems fundamentally exhibit situations where non-stationary optimization techniques (ones that optimize over time), are required.

2.1 Optimization over Time in Power Grid Engineering

Principally, optimal solutions can be computed in a stationary way for future time steps $t + K$, assuming that the near future can be predicted sufficiently. In such a case, K solutions can be obtained beforehand, considering variations of the system in the near-term interval, but expecting them to be deterministic and ignoring their potential stochastic nature.

However, in a dynamic and volatile system such as an electric power grid, it is more appropriate to make decisions as they come up. Thus, reacting to new order situations very quickly without computing a completely new solution at each time step t. This can be seen as the general aim of so called optimization over time.

Werbos [14] developed a class of methods to perform this kind of optimization in any engineering application, stating Aproximate Dynamic Programming (ADP) as a flexible and scalable technology being suitable for future smart grid issues as well. While classical Dynamic Programming is an exact method being limited by the so called "curse of dimensionality", ADP comes up with concepts of finding - as the name already says - sufficiently accurate approximate solutions to control problems [13]. Extending ADP, Werbos proposed a family of Adaptive Critic Designs (ACD), that combine ADP with reinforcement learing techniques for finding an optimal series of control actions that must be taken sequentially, not knowing their effects until the end of the sequence. While the standard ACD combines a set of artifical neural networks (ANN) within a neurocontroller for approximating the Hamilton-Jacobi-Bellman equation associated with optimal control theory, further concepts like Dual Heuristic Programming, Globalized Dual Heuristic Programming and Action-Dependent Heuristic Programming extend this formulation and build the family of ACDs.

Several researchers already picked up this idea of ACD, applying these concepts to electric power grid problems. Venayagamoorthy adapted it for building an automated voltage regulator of a generator system in [11] or even for operating a 12-bus power grid with an ACD-enabled Dynamic Stochastic Optimal Power Flow (DSOPF) controller in [7]. The necessity of solving such a DSOPF in future power grid operation has been substantiated by Momoh [8] [10] who proposed its generality and suitability to a large number of problems.

However all approaches related to ACD come from optimal control theory and thus try to approximate the Hamilton-Jacobi-Bellman equation with computational intelligence techniques. Building an alternative approach for optimization over time, the evolution of flexible control policies using metaheuristic algorithms as demonstrated within this work shows validity for electric power engineering problems as demonstrated further on, that does not necessitate a sequential decision process. Additionally, policies of less-complex and interpretable mathematical structure are evolved, rather than neurocontrollers composed of several ANNs and their interconnections as in ACDs.

2.2 Evolution of Flexible Control Policies

When optimizing power flows over time in a given distribution system, the aim is to find actions $u(t)$ of controllable units that lead to optimal behavior over a

given time interval $[t, (t+K)]$. In such a case, if all possible information of the system in near-future states $J(t+1)...J(t+K)$ could be predicted accurately, it would be possible to compute solutions (actions) $u(t+1)...u(t+K)$ beforehand that are optimal to each state, enabling optimal power grid operation in the near future. Here, steady-state optimization would be appropriate for finding an accurate solution to each state. But, what if the system changes at time $t+k$ with $k < K$? This event would cause the necessity of computing a completely new solution, making the usage of such a predictive steady-state optimization useless in dynamic systems. Here, it would be more appropriate to not compute all actions $u(t+1)...u(t+K)$ beforehand, but to make decisions online in a robust and quick way. Therefore, flexible policies can be used that are trained offline but lead to accurate decisions that react to new situations quickly, avoiding the need of re-optimization in each event while being able of considering full volatility and uncertainty of the system.

Figure 1 illustrates these two variants, namely the predictive computation of steady-state solutions in the upper part of the figure, versus the offline-training of a flexible policy, that takes system states at runtime and directly computes accurate actions online. While the first approach computes fixed solutions beforehand, that are unable to react to dynamic and uncertain conditions, the flexible policy-based approach keeps full flexibility during operation.

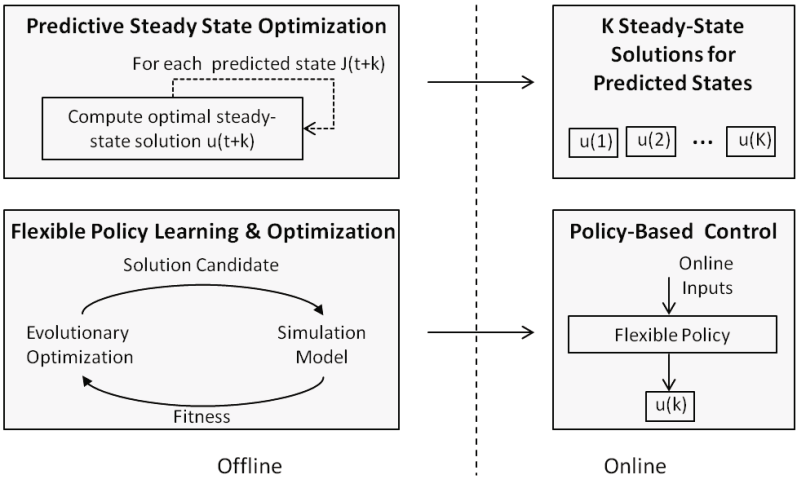


Fig. 1. Principles of Policy-Based Control

3 Policy Synthesis

Considering the steady-state solution to the general optimal power flow, the vector $\overline{u}$ typically consists of generator real power outputs (except the slack bus), generator voltages (including slack bus), shunt VAR compensators[1] as well as transformer

[1] Shunt-connected static reactive power (VAR) compensators.

tap settings. Hence, $\overline{u} = [P_2...P_{NG}, V_1...V_{NG}, Q_1...Q_{NVAR}, T_1...T_{NT}]$, with number of generator buses NG, number of VAR compensators $NVAR$ and the number of switchable transformers NT.

When using flexible policies, each variable (P,V,Q,T) is represented by a policy. This policy is a function of the system's global state $J(t)$ as well as the unit's local conditions and outputs the unit's actions $\overline{u(t)}$ in order to achieve the optimal power flow in the grid. This policy is principally the same for all controllable units of one variable (like for all shunt VAR compensators), but using information about the unit's specific situation, it leads to individual but globally optimal actions throughout the grid. Thus, the used information is crucial for building the policies since they have to represent the systems state sufficiently.

3.1 Construction of Atomic Rules

Tentatively, we assume a fixed mathematical structure $f(\overline{x})$ that represents the policy, where the vector $\overline{x}$ contains all state variables. Further, we assume that it is not necessary for the policy to consider the whole set of state variables, but to only use metrics that are important for power flow decisions. Thus, for these metrics so called "atomic rules" are introduced similar to [5], that gather necessary information from the system's state, supposing that this information is sufficient for achieving valid and accurate actions. All atomic rules are shown in Table 1.

This set of rules is considered to be necessary as well as sufficient in order to derive accurate actions for power flow control; the prove will be performed experimentally later. Here, the third column indicates which rules are necessary for learning a policy of a given variable. For example, the policy for variable P of a generator bus is a function $f(LLF, NLF, GLF, MARF, MERF, LCCF, QCCF)$. The application of atomic rules instead of using all state variables $\overline{x}$ is twofold: First, the cardinality of $\overline{x}$ is much smaller than the number of rules which is advantageous to the optimization of the rules. Second, the rules are completely independent of the grid's topology, making this approach applicable to any distribution or transmission grid model in a generic way.

Having the set of atomic rules $\overline{r}$, the policy $f(\overline{r})$ has to be computed in some way where a fixed mathematical structure is assumed which gets optimized by evolutionary algorithms. Since rules are normalized (by division with the maximum value in each rule), they can be integrated in any given mathematical structure.

3.2 Policy Synthesis and Optimization

In this work, two fixed mathematical structures are used for combining atomic rules and thus synthesizing the final policy out of them: a linear combination as well as a polynomial combination (for modeling eventual nonlinearities between rules). Using the linear combination of rules according to Equation 1 (variable P is shown as example), the set of weights has to be optimized when learning the policy for P, with NR being the cardinality of rules $\overline{r}$, i indicating the index of

Table 1. Atomic Rules

Rule	Explanation	Variable
LLF	Local Load Factor: active load at bus divided by maximum active power output at bus	P
NLF	Neighborhood Load Factor: sum of active load at directly connected buses and their neighbors divided by maximum active power output at those buses	P,V,Q,T
GLF	Global Load Factor: sum of total active load in grid divided by sum of maximum active power generation	P,V,Q,T
MARF	Max Rating Factor: maximum MVAR rating of connected branches divided by maximum MVAR rating of all branches	P,Q
MERF	Mean Rating Factor: mean MVAR rating of connected branches divided by maximum MVAR rating of all branches	P,Q
LCCF	Linear Cost Coefficient: linear cost coefficient of generator divided by maximum linear cost coefficient of all generators	P
QCCF	Quadratic Cost Coefficient: quadratic cost coefficient of generator divided by maximum quadratic cost coefficient of all generators	P
NRLF	Neighboring Reactive Load Factor: sum of reactive load at directly connected buses and their neighbors divided by maximum reactive power output at those buses	V,Q,T
GRLF	Global Reactive Load Factor: sum of total reactive load in grid divided by sum of maximum reactive power output	V,Q,T

the controllable unit. c additionally represents a constant that is evolved during the optimization process as well. Using the second representation as shown in Equation 2, policies are combined using polynomials of degree 2, thus, for each rule two weights ($nr, 1$ and $nr, 2$) are learned.

$$P_i = \frac{\sum_{nr=1}^{NR} r_{nr,i} * w_{nr}}{NR} + c \tag{1}$$

$$P_i = \frac{\sum_{nr=1}^{NR} r_{nr,i} * w_{nr,1} + r_{nr,i}^2 * w_{nr,2}}{NR} + c \tag{2}$$

Assuming the following policies to be optimized for all variables: $P(w_{LLF,P}$, $w_{NLF,P}$, $w_{GLF,P}$, $w_{MARF,P}$, $w_{MERF,P}$, $w_{LCCF,P}$, $w_{QCCF,P}$, $c_P)$, $V(w_{NLF,V}$, $w_{GLF,V}$, $w_{NRLF,V}$, $w_{GRLF,V}$, $c_V)$, $Q(w_{NLF,Q}$, $w_{GLF,Q}$, $w_{MARF,Q}$, $w_{MERF,Q}$, $w_{NRLF,Q}$, $w_{GRLF,Q}$, $c_Q)$ and $T(w_{NLF,T}$, $w_{GLF,T}$, $w_{NRLF,T}$, $w_{GRLF,T}$, $c_T)$, 21 real-valued weights ($\overline{w}$) and 4 constants ($\overline{c}$) need to be optimized, yielding 25 control variables (46 for the polynomial synthesis). In order to maintain bounding constraints for the variables P, V, Q and T, the output of the respective policy is multiplied by the variable's maximum value while guaranteeing that this value is not exceeded. Hence, policy optimization can be seen as regression problem.

3.3 Learning: Evolutionary Simulation Optimization

As already indicated in Figure 1, the learning of the policy is done offline by evolutionary optimization of the control variables. Since this optimization is performed within an uncertain system, namely a power grid dealing with stochastic and volatile load situations, a simulation model is chosen as appropriate system representation. This kind of simulation optimization has been studied extensively in [5] [6] [4] and enables the full integration of uncertain system behaviour into the optimization procedure. Thus, a solution candidate is represented by a real-valued vector, which is evaluated through simulation for expressing its fitness value. Figure 2 describes the decomposition of the real-valued solution vector into single policies.

$$\underbrace{P(\overline{r},\overline{w},c)}_{[w_{LLF}\ldots w_{QCCF}}\ \underbrace{V(\overline{r},\overline{w},c)}_{,w_{NLF}\ldots w_{GRLF}}\ \underbrace{Q(\overline{r},\overline{w},c)}_{,w_{NLF}\ldots w_{GRLF}}\ \underbrace{T(\overline{r},\overline{w},c)}_{,w_{NLF}\ldots w_{GRLF}]}$$

Fig. 2. Decomposition of the Solution Into Policies

4 Experimental Evaluation

In order to show the validity of the policy-based OPF approach as well as for fostering the reader's understanding of this method, it shall be demonstrated that it can solve the the general OPF for the IEEE 30-bus test system[2] [2]. This model consists of 30 buses and 41 branches, with the controllable units of 6 generator (PV) buses (including 1 slack bus) with variables P and V, 4 transformer tap changers with variable T and 9 shunt VAR compensators with variable Q. Solving the steady-state OPF, the solution vector $\overline{u}$ therefore has 24 variables. The formulation of the general OPF with all its constraints can be taken from the appropriate literature [15] [9] and would exceed the scope of this paper.

4.1 Training: Learning Optimal Policies

For modeling the optimal power flow control in a dynamic and uncertain environment based on this testcase, the power grid is simulated using Matpower[3] along a time-horizon of $K = 24$ hours, where the load changes over time according to a profile, but is additionally randomized in order to simulate real-world uncertain conditions. Within this simulation, the policies are trained such that they lead to optimal power flows within each possible simulated system state. For guaranteeing this, each solution is evaluated within 30 simulated power grid states that are spread along the time horizon in a probabilistic manner.

[2] `http://www.ee.washington.edu/research/pstca/`, Retrieved: 25.10.2012.
[3] `http://www.pserc.cornell.edu/matpower/`, Retrieved 25.10.2012.

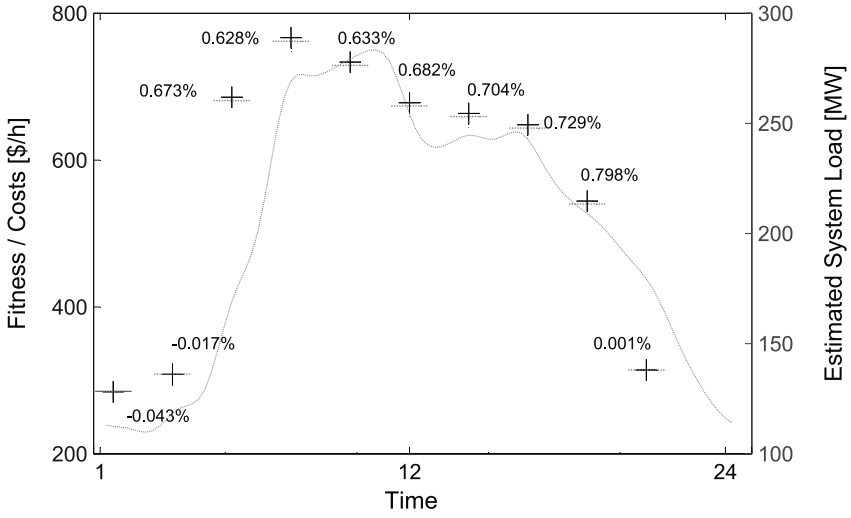


Fig. 3. Results: Exact Steady-State OPF vs. Policy Optimization over Time

As optimization algorithms, Evolution Strategies according to [3] are applied, since they are proven to perform well in real-valued optimization, also in simulation optimization for power grids [5] [6]. For the studies within this paper, finally (10+40)-ES has been proven to be successful, with discrete crossover and self adaptive manipulation described in [3]. This setting has been applied for all analysis herein, where the algorithm terminates after 3000 generations. Further information can be obtained from the documentation of the HeuristicLab [4] software [1], which is the core framework for our investigations.

4.2 Testing: Evaluating the Learned Policies

In order to evaluate the validity of the learned flexible policies on a separate test set, ten discrete system states spread over the day are expressed from the simulation. For these states, the exact steady-state solution to the OPF is computed with primal dual interior point method, implemented in MatPower. This solution is then compared to the solution that the policies would lead to in this state, shown in Figure 3. In this diagram, the fitness of the exact solution (grey dottel horizontal lines) is plotted with respect to the fitness of the best found policy's output (black plus-signs), where the relative difference is indicated for each discrete state. On the right-hand ordinate, the electric load of the system is shown along the day, represented by the dotted line. Since uncertainties are modeled in the simulation, this line only indicates the mean estimated load profile,

[4] HeuristicLab, a paradigm-independent and extensible environment for heuristic optimization, `http://dev.heuristiclab.com/`

which is randomized in each sampled state. In order to prove the approach on a volatile system, the variance of the load over the day as well as its uncertainty is taken to be much higher than in typical real-world scenarios.

For these then test-states, an impressive result can be stated: the outputs of the flexible policy are competitive with the exact solutions computed separately for each state, with an average relative difference in fitness of 0.48%. Especially for two low-load states in the early morning, the policy outputs even perform slightly better. This may occur because the interior-point method uses a linearized model of the power grid for optimization, which is only an approximation, while the simulation optimization of policies uses the actual power flow model.

The results clearly demonstrate the power of this approach, where a flexible policy is learned offline and avoids any re-optimization during operation of a volatile and uncertain power grid. These best found policies are based on polynomial synthesis as described above, where the best found policies with linear synthesis perform in average 1% worse in means of fitness, but nevertheless are competitive to the exact solutions.

5 Conclusion

A novel method has been introduced that provides a suitable approach for power flow optimization over time. It eliminates the necessity of computing the exact steady-state solution to the OPF in each situation of a dynamic and uncertain system, while intelligent policies deliver accurate actions quickly and robust at runtime. These policies are learned offline using heuristic simulation optimization with Evolution Strategies, which enables full integration of the dynamic and probabilistic system behaviour into the optimization procedure. Here, the application to the general OPF has been demonstrated for the IEEE 30-bus test system for proving its validity, but numerous additional applications lie on hand like shown in [5], that potentially benefit using the discussed principles. For example, with the same approach policies could be evolved for intelligent and autonomous error recovery after system outages, or even policies for optimal unit commitment in a volatile microgrid, to name just a few. Thus, a generic, flexible and scalable approach was presented that is capable of representing a fundamental tool for future smart electric grids.

Acknowledgment. This project was supported by the program Regionale Wettbewerbsfähigkeit OÖ 2010-2013, which is financed by the European Regional Development Fund and the Government of Upper Austria, as well as the "Clean Motion Offensive" Project (FFG-Nr.: 829100).

References

1. Affenzeller, M., Wagner, S., Winkler, S., Beham, A.: Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. CRC Press (2009)

2. Alsac, O., Stolt, B.: Optimal load flow with steady-state security. IEEE Transactions on Power Apparatus and Systems PAS-93(2), 745–751 (1974)
3. Beyer, H.G., Schwefel, H.P.: Evolution strategies - a comprehensive introduction. Natural Computing 1 (2002)
4. Fu, M.C.: Feature article: Optimization for simulation: Theory vs. practice. INFORMS Journal on Computing 14, 192–215 (1977)
5. Hutterer, S., Auinger, F., Affenzeller, M.: Evolutionary optimization of multi-agent control strategies for electric vehicle charging. In: Companion Publication of the 2012 Genetic and Evolutionary Computation Conference (2012)
6. Hutterer, S., Auinger, F., Affenzeller, M.: Metaheuristic optimization of electric vehicle charging strategies in uncertain environment. In: International Conference on Probabilistic Methods Applied to Power Systems (2012)
7. Liang, J., Venayagamoorthy, G.K., Harley, R.G.: Wide-area measurement based dynamic stochastic optimal power flow control for smart grids with high variability and uncertainty. IEEE Transactions on Smart Grid 3, 59–69 (2012)
8. Momoh, J.A.: Toward dynamic stochastic optimal power flow. In: Si, J., Barto, A., Powell, W., Wunsch, D. (eds.) Handbook of Learning and Approximate Dynamic Programming, pp. 561–598. Wiley Interscience (2004)
9. Momoh, J.A.: Electric Power System Applications of Optimization, 2nd edn. CRC/Taylor & Francis (2009)
10. Momoh, J.A., Zivi, E.: Control, optimization, security, and self-healing of benchmark power systems. In: Si, J., Barto, A., Powell, W., Wunsch, D. (eds.) Handbook of Learning and Approximate Dynamic Programming, pp. 599–637. Wiley Interscience (2004)
11. Venayagamoorthy, G.K., Harley, G., Wunsch, D.: Applications of approximate dynamic programming in power systems control. In: Si, J., Barto, A., Powell, W., Wunsch, D. (eds.) Handbook of Learning and Approximate Dynamic Programming, pp. 479–515. Wiley Interscience (2004)
12. Wang, H., Murillo-Sánchez, C.E., Zimmerman, R.D., Thomas, R.J.: On computational issues of market-based optimal power flow. IEEE Transactions on Power Systems 22(3), 1185–1193 (2007)
13. Werbos, P.J.: Adp: Goals, opportunities and principles. In: Si, J., Barto, A., Powell, W., Wunsch, D. (eds.) Handbook of Learning and Approximate Dynamic Programming, pp. 3–44. Wiley Interscience (2004)
14. Werbos, P.J.: Computational intelligence for the smart grid - history, challenges, and opportunities. IEEE Computational Intelligence Magazine 6(3), 14–21 (2011)
15. Wood, A.J., Wollenberg, B.F.: Power Generation, Operation, and Control, 2nd edn. Wiley Interscience (1996)

Using a Genetic Algorithm for the Determination of Power Load Profiles

Frédéric Krüger[1], Daniel Wagner[2], and Pierre Collet[1]

[1] Université de Strasbourg, ICube Laboratory
[2] Électricité de Strasbourg Réseaux
{frederic.kruger,pierre.collet}@unistra.fr,
daniel.wagner@es-groupe.fr

Abstract. Electrical distribution companies struggle to find precise estimations of energy demand for their networks. They have at their disposal statistical tools such as power load profiles, which are however usually not precise enough and do not take into account factors such as the presence of electrical heating devices or the type of housing of the end users. In this paper, we show how a genetic algorithm generated with the EASEA language can be successfully applied to solve a noisy blind source separation problem and create accurate power load profiles using real world data provided by "Électricité de Strasbourg Réseaux". The data includes load measurements of 20kV feeders as well as the energy consumption of more than 400,000 end users. The power load profiles obtained demonstrate considerable improvement in the estimation of load curves of 20kV feeders.

1 Introduction

Accurate predictions of the energy demand of a small group of low-voltage consumers are very difficult to achieve. Power load profiles have been designed to address this problem. Load profiles are statistical tools that model the energy consumption of a certain class of customers for a given half-hour of the day. The issue with these load profiles is that they are often imprecise as they do not take into account important external factors such as the end user's heating devices or their type of housing (single house or apartment). Blanking out these factors results in the creation of inaccurate load curves for areas very sensitive to temperature changes (for example areas with a great amount of electrical heating devices, very sensitive to small temperature variations in the winter time). Predicting the energy demand accurately remains a challenge electrical distribution keep struggling with.

The determination of load profiles is a very prolific area of research [1–8]. But it seems that most of the articles addressing the problem are related to series analysis of measurements of medium voltage to low voltage (MV-LV) substations or low voltage (LV) end users. The measurement series are usually obtained through very time consuming and expensive measurement campaigns. A lot of examples simply apply data mining techniques on individual end user load

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 162–171, 2013.
© Springer-Verlag Berlin Heidelberg 2013

curve measurements [1–3]. Sometimes external factors such as the outside temperature are taken into account to study their influence on the shape of the load profiles [4]. Some studies show how the generation of profiles can be combined with customer segmentation [5]. Some authors propose a bottom-up approach combining socio-economic as well as demographic end user information along with individual household appliance load profiles to determine end user profiles through simulations [6, 7]. The issue with measurement time series is that as they are very time consuming and expensive, they are very difficult to acquire making the creation of new load profiles tedious with the absence of such information. Yet, a paper of 2003 [8] demonstrates that it is possible, without any prior knowledge of the electrical network and without costly measurement series, to find load profiles. The authors handle the problem as a blind source separation and apply their method on artificial datasets. In this paper we propose a similar method to [8] to determine specific power load profiles without measurement campaigns, only using knowledge from a real world electrical network and an evolutionary algorithm.

2 General Setup

This section gives information regarding the structure of the electrical network operated by Électricité de Strasbourg Réseaux (ÉSR), the data available in this study, as well as the process applied for the creation of the load profiles.

2.1 Available Data

The data accessible during this research was supplied by ÉSR. ÉSR is an energy distribution company that provides electricity to more than 440,000 low voltage as well as high voltage end users. ÉSR maintains and operates a network of about 14,000 km.

Structure of the Electrical Network. The network is structured in different sub-networks:

1. **The high voltage sub-network.** This meshed network has a voltage of 63kV. It includes several 63kV/20kV high voltage to medium voltage (HV-MV) substations. Each substation feeds several hundreds distribution stations connected through 20kV feeders to the medium voltage network. The ÉSR high voltage network has about 40 substations.
2. **The medium voltage sub-network.** This tree-shaped network has a voltage of 20kV. It feeds 20kV/400V MV-LV substations connected to the low voltage network or to high voltage end-users. The ÉSR medium voltage network has about 7000 MV-LV substations.
3. **The low voltage sub-network.** This tree-shaped network delivers power to end users. A MV-LV substation (the root of the sub-network) delivers electricity to about 100 end users in average.

Each 20kV feeder is equipped with a measurement device that records its average load at a 10mn interval. The resulting load curves are stored in a database. ÉSR also records each end user's energy consumption through meter readouts over 6 months intervals.

Types of End Users. Load profiles embody the load curve of a category of end users. The end users have hence to be separated into different groups. In this study, the end users are separated into 8 classes following specific criteria:

1. Use: domestic or professional
2. Type of heating: with or without electrical heating
3. Type of housing: apartment or house
4. Tariff: single or double rate tariff (offpeak/peak hours)

The classes identified in table 1 were formed through statistical analysis of the information found in the different databases concerning the end users. Machine learning techniques were applied to detect the presence of electrical heating allowing the separation of the end users in 2 groups with/without electrical heating.

Table 1. Different classes of end users

Use	With/Without Electrical heating	Type of housing	Tariff	Relative amount
Professional	With	-	-	4%
Professional	Without	-	-	10%
Domestic	With	House	-	7%
Domestic	Without	House	Single rate	18%
Domestic	Without	House	Double rate	8%
Domestic	With	Apartment	-	15%
Domestic	Without	Apartment	Single rate	31%
Domestic	Without	Apartment	Double rate	7%

2.2 Methodology

We make the following assumption: the load curve of a single feeder contains the load curve and hence the profiles of the end users connected to that feeder. The resulting problematic is very close to a blind signal separation [9]: separate a set of signals (in our case a set of 8 profiles) from a set of mixed signals (in our case a set of feeder load curves). We assume that the different profiles are mutually independent. Using that approach, the electrical network must be virtually reconstructed and every end user has to be correctly connected to its HV-MV feeder. This compels us to verify that the topology of the electrical network used is accurate.

As a matter of fact, the electrical network has a nominal topology that can be subject to short localized changes as well as long term changes as (for instance, in

the case of new constructions). These changes in the network topology necessarily impact the shape of the load curves. Failures of a feeder's measuring device can also lead to errors and irregularities in the load curves. In order to successfully perform the signal separation we need to make sure that the set of mixed signals *i.e.* the feeder load curves are not subject to topology changes of the MV network or measurement errors.

The verification is achieved by comparing the energy W_m (in kWh) distributed by a feeder d during the period of time t with the energy W_c (in kWh) consumed by all the end users connected to d. For the time period t we chose a time span of one month. That time span matches the energy measurement cycles of HV end users. The energy consumption of the LV end users is calculated by estimating a monthly energy from their biannual energy readings.

For every month of a year and every feeder, we perform the following comparison

$$W_m = W_c + \epsilon \tag{1}$$

The value ϵ can be explained by a lot of factors such as energy loss (Joule's law), monthly energy estimation errors (when computing the LV end user's monthly energy), measurement device failure, network topology changes, errors in the databases, etc... The factor that influence the most the ϵ value are network topology changes. The greater ϵ for a feeder d, the greater chances are that the topology used to reconstruct d's sub-network is imprecise. Keeping only the load curves of the distribution stations which energy minimizes ϵ ensures us that the load curves in the mixed signal set is free from "excessive" external noise. Unfortunately, we cannot be too demanding with the quality of the load curves. By keeping load curves which ϵ value ranges between -10% and 20%, we manage to keep between 10% and 15% of the load curves.

To perform the blind signal separation, we chose to use a genetic algorithm. These metaheuristic optimization algorithms have proven being very efficient in solving this class of problems [10, 11]. The challenge in the case of our study is to overcome the noise caused by the load curves with a great ϵ value, but the authors of [12] showed that genetic algorithms are more than capable of solving this kind of noisy problems.

3 Genetic Algorithm

Genetic Algorithms (GA) are population-based optimization algorithms belonging to the larger class of evolutionary algorithms (EA) . They are inspired by biological evolution and Darwin's theories of the survival of the fittest individual.

3.1 Basic Genetic Algorithm

A genetic algorithm strives to find the optimal solution to a problem by evolving a population of potential solutions for a certain amount of iterations (generations). The basic evolutionary algorithm [13] creates a random population of

individuals, evaluates their fitness and starts the evolution process. Individuals from the initial "parent" population are selected with a bias towards the best to create "children" via crossover and mutation functions that combine the individuals' genetic material. The fresh children population is then evaluated as well. Finally another selection operator takes the best from parents and children to create the population for the next generation. This process is usually repeated until the maximum number of generations is achieved.

3.2 EASEA

The genetic algorithm implemented to find the load profiles was generated with EASEA (EAsy Specification of Evolutionary Algorithms)[1]. EASEA is a language and platform that allows an easy specification of an evolutionary algorithm [14]. Only problem specific pieces of code, *i.e.* individual representation, initialization, crossover, mutation and fitness function, need to be supplied in an *.ez* file that also contains the parameters for the evolutionary engine. EASEA can generate a multitude of different evolutionary paradigms such as (GA, GP, memetic algorithms, island models, etc) by the means of the same specification file. A unique feature of the EASEA language is its ability to parallelize the fitness function on a GPGPU card, making it possible to achieve great speedups and solving problems a lot faster.

3.3 Fitness Function

The fitness evaluation function is a mix of two different functions: a function that compares the estimated curve with the measured curve and a function that measures the irregularity of the load profiles. The following equation allows the genetic algorithm to rate, for a single distribution station c, the difference between the measured load curve and the estimated load curve using an individual *i.e.* a set of profiles

$$F_c = \frac{\sum_{t=t_0}^{t_1} |p(t) - (W_1 P_1(t) + \ldots + W_n P_n(t))|}{p(t)} \tag{2}$$

where

- $p(t)$ load of the distribution station at time t (in kW);
- W_n average power of end users with profile n (in kW);
- $P_n(t)$ value of profile n at time t (no unit);

The total fitness F for an individual is given over the complete set of m load curves

$$F - \frac{\sum_{c=1}^{m} F_c + \sum_{i=1}^{n} MAX((d - D_i), 0)}{n + 1} \tag{3}$$

[1] `http://sourceforge.net/projects/easea`

where

- m number of measured load curves in the dataset;
- n number of profiles (10 in the case of this study);
- d overall average distance between two half-hour points in a profile;
- D_i average distance between two half-hour points of profile i;

The genetic algorithm must find the best individual *i.e.* the best set of profiles that minimizes the fitness value F. The first part of the fitness function, the distance between measured and estimated load curve, is designed to make the genetic algorithm find the basic shape of the profiles. The second part of the fitness function, the distance between the half-hour points of the profiles, is designed to install regularity within the profiles and smooth their shape.

3.4 Engine of the Genetic Algorithm

Initialization Function. The initialization function commonly generates the individuals randomly. We chose to implement a problem specific initialization function based on the average distance between each profile half-hour point. The first half-hour coefficient of each profile is initialized randomly. Then, the value of each subsequent coefficient is given by the following function

$$p[i] = p[i-1] + random(-\frac{MaxCoeffDistance}{2}, \frac{MaxCoeffDistance}{2}) \quad (4)$$

The problem specific initialization function allows the GA to converge a lot faster than the random initialization. Figure 1 shows that the GA with the random initialization requires 2000 generations to achieve the same fitness where the GA with the custom initialization requires less than 1000 generations. What Figure 1 does not clearly reveal is that the GA with the custom initialization returns results with better fitness at the end of the 3000^{th} generation.

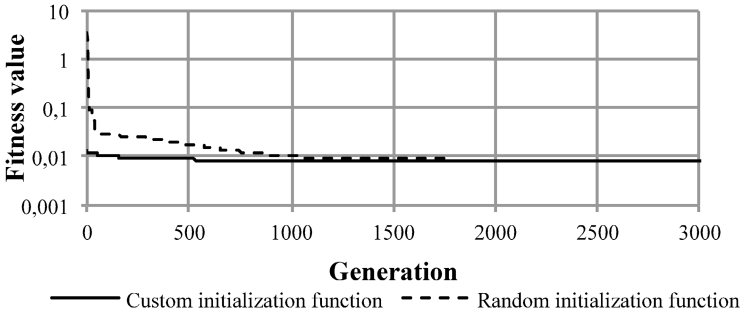


Fig. 1. Comparing the average convergence over 30 runs of the GA with the custom initialization function and the GA with the random initialization function

Genotype Depiction. An individual represents a set of 8 profiles for a specific day in the month. Each profile is represented as an array of 48 floating points (one for every half-hour in the day). Therefore, the genotype of an individual is a 48×8 long floating point array.

GA Parameters. As a mutation function we chose a simple addition of noise to a random coefficient of a profile. A coefficient has a probability of 0.005 to be mutated. An average of 0.5% genes is being mutated. As a crossover function we chose to implement a barycentrical crossover.

Table 2 gives the parameters used to determine the load profiles. These parameters where determined through a selection process based on computation time and fitness quality.

Table 2. Evolutionary engine

Parameter	Value
Number of generations	6000
Population size	750
Number of offspring	500
Selection operator	Tournament
Selection pressure	20
Reduction operator	Tournament
Reduction pressure	10

4 Results

4.1 Load Profiles

The evolutionary algorithm computes the load profiles on a daily basis meaning that, for a given month, the GA optimizes each day of the month independently of the other. As a result a set of profiles for each day of a given month is obtained. Figure 2 shows 3 of the 8 profiles returned by the GA for a day of January 2009.

The dotted curve in figure 3 shows the profile of professional end users without electrical heating with a typical business time load (peak hours between 8am-12am and 2pm-5pm).

The lined curve shows the profile of domestic end users without electrical heating living in apartments. We can see that the curve has a large load dynamic with $\frac{Max-Min}{Avg} \approx 150\%$ and the peak around 7pm.

Finally, the dashed curve shows the profile of domestic end users with electrical heating living in apartments. The curve has a flatter load dynamic with $\frac{Max-Min}{Avg} \approx 60\%$.

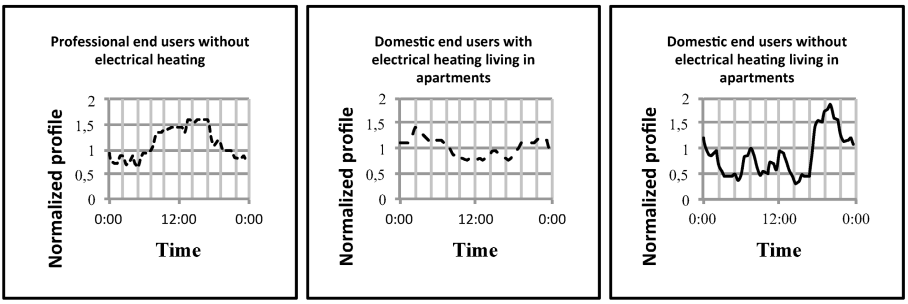


Fig. 2. Example of 3 profiles found by the genetic algorithm

4.2 Load Curve Estimation

The profiles found by the genetic algorithm have been approved by an expert and validated as coherent with reality. Nevertheless, their quality was checked by comparing a) the load curve estimated by using the GA profiles with b) the load curve built with the profiles used before this study, and c) the measured load curve of the MV feeders.

Figures 3 and 4 show the comparison performed for two feeders. Feeder d_1 has a greater proportion of end users living in apartments with electrical heating (99% of the 31% of end users with electrical heating), whereas feeder d_2 has a greater proportion of end users living in independent households with electrical heating (60% of the 25% of end users with electrical heating). The two figures clearly reveal that the estimated load curve built using the optimized profiles match the measured load curve. The dotted curves completely underestimate the energy demand due to the electrical heating. The figures also reveal that the shape of the two estimated load curves are completely different.

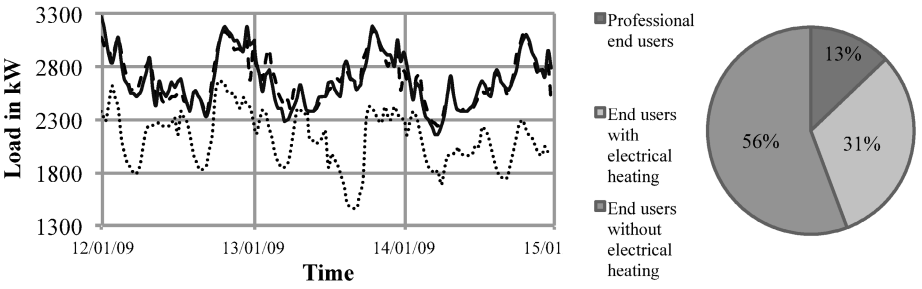


Fig. 3. Feeder d_1. The chart on the right side shows the end user proportions (simplified), the chart on the left side the different load curves: measured load curve (lined curve), estimated load curve using the original profiles (dotted curve) and estimated load curve using the optimized profiles (dashed curve).

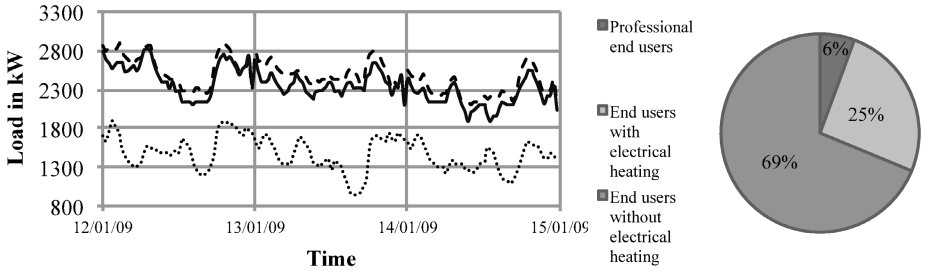


Fig. 4. Feeder d_2. The chart on the right side shows the end user proportions (simplified), the chart on the left side the different load curves: measured load curve (lined curve), estimated load curve using the original profiles (dotted curve) and estimated load curve using the optimized profiles (dashed curve).

Table 3 and 4 show the results for two quality metrics: correlation coefficient to verify the similarity between the shape of the curves and the root mean square error (RMSE). Table 3 compares measured curves and estimated curves for feeder d_1 and table 4 compares measured curves and estimated curves for feeder d_2. The close to 1 correlation coefficient values as well as the low RMSE values attest of the quality of the profiles found by the GA.

Table 3. Quality metrics for feeder d_1

Quality quantification	Measured load curve *vs*	
	Estimated load curve using **GA** profiles	Estimated load curve using **old** profiles
Correlation Coefficient	0.92	0.51
RMSE (kW)	91.12	608.62

Table 4. Quality metrics for feeder d_2

Quality quantification	Measured load curve *vs*	
	Estimated load curve using **GA** profiles	Estimated load curve using **old** profiles
Correlation Coefficient	0.94	0.53
RMSE (kW)	144.08	901.10

5 Conclusion

In this study we show how genetic algorithms can be successfully used to find high quality load profiles without expensive measurement campaigns and successfully solve the signal separation problem with a genetic algorithm. Including external factors such as electrical heating or type of housing in the design of the load profiles enhances drastically the accuracy of the estimated load curves. This paper paves the way for much more accurate predictions of load curves at the local network level for instance MV/LV transformers.

Acknowledgments. The authors gratefully acknowledge the financial contribution of "Région Alsace", as well as the contribution of G. Arnold for his work in the field of electrical heating detection.

References

1. Jardini, J.A., Tahan, C.M., Gouvea, M.R., Ahn, S.U., Figueiredo, F.M.: Daily Load Profiles for Residential, Commercial and Industrial Low Voltage Consumers. IEEE Transactions on Power Delivery 15(1), 375–380 (2000)
2. Gerbec, D., Gasperic, S., Smon, I., Gubina, F.: Allocation of the Load Profiles to Consumers Using Probabilistic Neural Networks. IEEE Transactions on Power Systems 20(2), 548–555 (2005)
3. Chen, C.C., Wu, T.H., Lee, C.C., Tzeng, Y.M.: The Application of Load Models of Electric Appliances to Distribution System Analysis. IEEE Transaction on Power Systems 10(3), 1376–1382 (1995)
4. Chen, C.S., Kang, M.S., Hwang, J.C., Huang, C.W.: Temperature Effect to Distribution System Load Profiles and Feeder Losses. IEEE Transactions on Power Systems 16(4), 916–921 (2001)
5. Espinoza, M., Joye, C., Belmans, R., De Moor, B.: Short-Term Load Forecasting, Profile Identification, and Customer Segmentation: A Methodology Based on Periodic Time Series. IEEE Transactions on Power Systems 20(3), 1622–1630 (2005)
6. Figueiredo, V., Rodriguez, F., Vale, Z., Gouveia, J.B.: An Electric Energy Consumer Characterization Framework Based on Data Mining Techniques. IEEE Transactions on Power Systems 20(2), 596–602 (2005)
7. Capasso, A., Grattieri, W., Lamedica, R., Prudenzi, A.: A Bottom-Up Approach to Residential Load Modeling. IEEE Transactions on Power Systems 9(2), 957–964 (1994)
8. Liao, H., Niebur, D.: Load Profile Estimation in Electric Transmission Networks Using Independent Component Analysis. IEEE Transactions on Power Systems 18(2), 707–715 (2003)
9. Acharyya, R.: A New Approach for Blind Source Separation of Convolutive Sources (2008)
10. Rojas, I., Clemente, R.M., Puntonet, C.G.: Nonlinear Blind Source Separation Using Genetic Algorithms. In: Independent Component Analysis and Signal Separation (2001)
11. Shyr, W.J.: The Hybrid Genetic Algorithm for Blind Signal Separation. In: Neural Information Processing, pp. 954–963 (2006)
12. Katou, M., Arakawa, K.: Blind Source Separation in Noisy and Reverberating Environment Using Genetic Algorithm. In: Proceeding of 2009 APSIPA Annual Summit and Conference (2009)
13. De Jong, K.: Evolutionary computation: a unified approach. In: Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference, GEGGO (2008)
14. Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take it EASEA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 891–901. Springer, Heidelberg (2000)

Comparing Ensemble-Based Forecasting Methods for Smart-Metering Data

Oliver Flasch, Martina Friese, Katya Vladislavleva, Thomas Bartz-Beielstein, Olaf Mersmann, Boris Naujoks, Jörg Stork, and Martin Zaefferer

Fakultät für Informatik und Ingenieurwissenschaften, FH Köln
Steinmüllerallee 1, 51643 Gummersbach
{First Name.Last Name}@fh-koeln.de,
katya@evolved-analytics.com

Abstract. This work provides a preliminary study on applying state-of-the-art time-series forecasting methods to electrical energy consumption data recorded by smart metering equipment. We compare a custom-build commercial baseline method to modern ensemble-based methods from statistical time-series analysis and to a modern commercial GP system. Our preliminary results indicate that that modern ensemble-based methods, as well as GP, are an attractive alternative to custom-built approaches for electrical energy consumption forecasting.

1 Introduction

Smart metering equipment records electrical energy consumption data in regular intervals multiple times per hour, streaming this data to a central system, usually located at a local public utility company. Here, consumption data can be correlated and analyzed to detect anomalies such as unusual high consumption and to provide energy consumption forecasts.

This paper describes results from an ongoing project with GreenPocket GmbH (`http://greenpocket.de`), a software provider for smart metering infrastructure. GreenPocket's software aims at giving consumers insight into their consumption habits and provide them with accurate forecasts of their future energy consumption. Among GreenPocket's customers are electric utility companies who collect and analyze smart metering data of thousands of customers. This means that the energy consumption forecasting methods employed have to be scalable and efficient. Since the forecast models applied by GreenPocket are relatively simple, it is of great interest to compare them with more sophisticated modeling approaches, including symbolic regression (SR) and ensemble-based model selection. From our experience, both of these approaches provide very good accuracy and scalability. We present a preliminary experimental study based on real-world data to analyze the applicability of these approaches in a real-world setting. At this stage we focus on comparing modern existing approaches.

Research goals of this study will be presented in Section 2, while Section 3 describes data and experiments. The different prediction methods are introduced in Section 4 and Section 5 discusses the results. The paper concludes with a short outlook in Section 6.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 172–181, 2013.
© Springer-Verlag Berlin Heidelberg 2013

2 Research Goals

The first goal of our study is to analyze benefits of ensemble-based approaches, i.e., approaches that evaluate several time-series models in parallel and allow an adaptation or even change of the current model based on new features and trends in the data. In theory, model adaptation improves the prediction accuracy. The second goal of our study is to analyze the performance of symbolic regression, a generic modeling approach, on the same real-world problem of electrical energy consumption forecasting. This application is based on genetic programming (GP) and not limited to time-series forecasting.

This study will investigate the following research questions:

- **Q1** Can modern ensemble-based approaches, without further domain knowledge about the data used, compete with custom-built approaches in a real-world energy consumption time-series prediction scenario?
- **Q2** Is symbolic regression, as a generic method, able to create time-series prediction models as accurate as custom-built approaches in the same scenario?

3 Data and Experiments

All models are trained on the same data set and are then used to predict the same time interval. The experiments are run on energy consumption time-series data supplied by GreenPocket. The data was recorded by two independent smart metering devices, installed at a commercial customer. Some data points are missing due to measurement or transmission issues, which is a common situation in real-world settings.

3.1 Training- and Test-Datasets

The data provided by GreenPocket are series of timestamp and meter reading pairs taken quarter-hourly, a standard frequency for modern smart metering equipment. Timestamps are given an ISO 8601 derived date/time format, meter readings are given in kilowatt hours (KWH). As the energy consumption time-series data was recorded by two independent smart metering devices (`meter1` and `meter2`), each split into two time intervals (`series1` and `series2`) this results in four data sets.

We focus on the first time-series data set, i.e., `meter1_series1`, which was split into a training and a test data set for our experiments. The training interval of this data set starts at `2010-12-06 23:15:00` and ends at `2011-03-06 00:00:00`, the test interval starts at `2011-03-06 00:15:00` and ends at `2011-04-04 21:45:00`. This amounts to a training data length of approximately 12 weeks and a test data length of 4 weeks. Figure 1 shows plots of this training data set. Note that there are missing data points in the training data set. Visual inspection reveals daily periods, while weekly periods are detectable, but not as clearly defined. To give a first assessment of the generalizability of our ensemble methods, we also conducted first experiments with `meter1_series2` data.

3.2 Prediction Quality Rating

For these experiments each approach was fitted on the training data introduced above and had to deliver a quarter-hourly prediction time series for the 4 weeks of test data

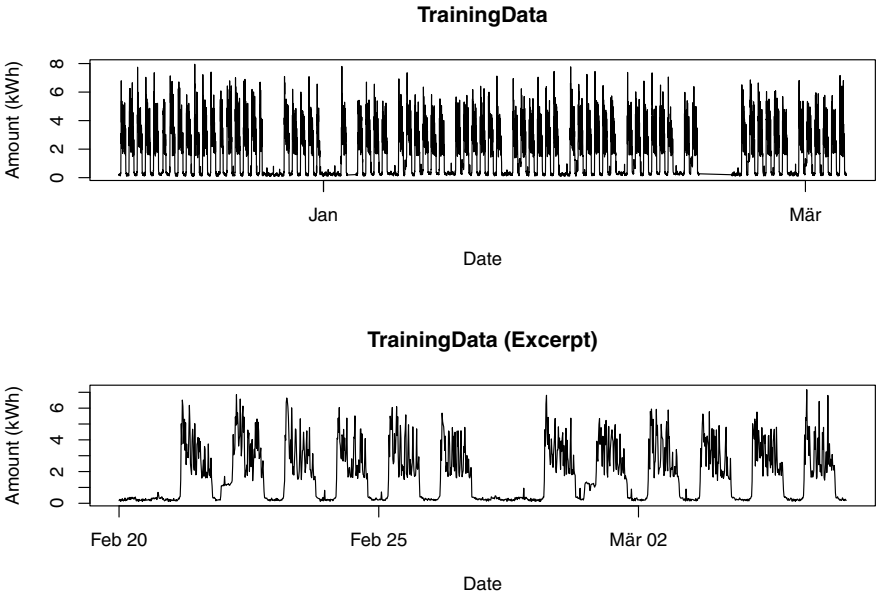


Fig. 1. Plots of the training time series `meter1_series1`. Energy consumption meter readings (measured in KWH) were recorded every fifteen minutes. The upper plot shows the complete data range, the lower plot the last two weeks.

given above. To evaluate the quality of a predicted electrical energy consumption time series we consider three different error measures.

MAE. The *mean absolute error* (MAE) between the predicted time series $\hat{t}$ and the respective true energy consumption (test) time series t. Equation 1 defines the RMSE.

$$\text{MAE}(\hat{t}, t) := \frac{\sum_{i=1}^{n} |\hat{t}_i - t_i|}{n} \tag{1}$$

RMSE. The *root mean square error* (RMSE) between the predicted time series $\hat{t}$ and the respective true energy consumption (test) time series t. Equation 2 defines the RMSE.

$$\text{RMSE}(\hat{t}, t) := \sqrt{\frac{\sum_{i=1}^{n} (\hat{t}_i - t_i)^2}{n}} \tag{2}$$

RMSElog. The *root mean square error log* (RMSElog) is the RMSE between the logarithm of the predicted time series $\hat{t}$ and the logarithm of the respective true energy consumption (test) time series t:

$$\text{RMSElog}(\hat{t}, t) := \sqrt{\frac{\sum_{i=1}^{n} (\log(1 + \hat{t}_i) - \log(1 + t_i))^2}{n}}. \tag{3}$$

We apply a logarithmic transformation because energy consumption is always positive or zero and its distribution is highly skewed. The RMSE on the other hand is a symmetric loss function and therefore is best applicable if the error distribution is symmetric. By applying a log-transformation we hope to obtain a less skewed distribution. We also postulate that this loss is closer to what is important in a practical application.

4 Methods

The ensemble-based methods that we focus on in this work are implemented in the R forecast package and include Exponential smoothing state space (ETS) models and automated Autoregressive Integrated Moving Average (ARIMA) modeling [1]. Using ensembles the software chooses a single model from the large model class, which is used for prediction. In addition, we study the performance of symbolic regression via GP as a general method not limited to univariate time-series forecasting. The symbolic regression [2] implementation we use is DataModeler, a commercial GP package based on Mathematica. All methods where applied with their default parameter settings.

4.1 Baseline (GreenPocket)

The baseline method provided by GreenPocket and applied in their production systems is an example of an extremely simple yet computationally highly efficient time-series prediction method. The prediction is the average energy consumption of the last 14 days at the same time of day as the prediction. This model is even simpler than a moving average as it has a fixed time horizon (14 days) after which an observation has no influence on a prediction.

4.2 Ensemble-Based Methods

Classical time-series forecasting methods, including exponential smoothing state space models or ARIMA models, are widely and successfully applied to practical time-series forecasting problems much like the one discussed in this work. Both ETS and ARIMA models can capture a wide variety of different data generating processes. Consequently it is the burden of the user to choose a set of parameters for the model such that it adequately fits the data. Because selecting an appropriate model structure for a given forecasting problem is essential for obtaining good results, this selection process is often considered difficult by users not trained in statistical time-series analysis. Furthermore, manual model structure selection is a time-consuming and error prone task even for trained users.

To alleviate these difficulties and to enable automatic forecasting for time-series data of unknown structure, ensemble-based methods have been developed that automate the model selection process. In this work, we study the accuracy of two state-of-the art methods for automatic time-series forecasting: Automated ARIMA models and automated exponential smoothing state space models.

Both methods are limited to time-series with small to medium-sized seasonal frequencies. The automated ARIMA implementation we use in this study is able to support seasonal period lengths of up to $m = 350$ time units, while in practice, memory

constraints of our implementation will limit this value to about $m = 200$. Yet, in theory, the number of parameters to be estimated during ARIMA model fitting does not depend on m, so any m should be possible. Similarly, the automated ETS implementation we use restricts seasonality to a maximum period of 24 time units. This limitation stems from the fact that there are $m - 1$ parameters to be estimated for the initial seasonal states. As model parameters have to be estimated for many models structures, the automated ETS algorithm becomes computationally infeasible for large m.

As mentioned in Section 3, our quarter-hourly training data (96 measurements per day) indicates daily as well as weekly periods, warranting a period length of $m = 672$ to capture the weekly periodicity in the data. Unfortunately, this puts this problem clearly out of reach of our current implementations of both automated ARIMA and automated ETS. As a simple workaround, we therefore applied the STL decomposition to seasonally adjust the data, only then applied automated ETS or automated ARIMA to forecast the adjusted data, and finally added the seasonal component back into the forecasts. STL is a procedure that decomposes a seasonal time-series into trend, seasonal, and remainder components by a sequence of applications of the LOESS smoother [3]. We used the STL implementation from the R stats package [4].

Automated ARIMA *Models.* By using STL to seasonally adjust the input data, we are able to apply a non-seasonal ARIMA(p, d, q) process of the form

$$\phi(B)(1 - B)^d y_t = c + \theta(B)\epsilon_t. \tag{4}$$

Here, $\{\epsilon_t\}$ is a white noise process with mean zero and variance σ^2. B is the backshift operator, and $\phi(z)$ and $\theta(z)$ are polynomials of order p and q. For $c \neq 0$, the implied polynomial in the forecast function has order d. Automated ARIMA forecasting then consists of selecting appropriate values p, q and d, i.e. an appropriate model order. We do this using Akaike's Information Criterion (AIC).

$$\text{AIC} := -2\log(L) + 2(p + q + k), \tag{5}$$

where $k := 1$ if $c \neq 0$ and 0 otherwise. L is the likelihood of the model when fit to the differenced data $(1 - B)^d y_t$. As the likelihood of the full model for y_t is not actually defined and therefore AIC values for different levels of differencing are not comparable, d is chosen via unit-root tests based on a null hypothesis of no unit-root. ARIMA(p, d, q) models where d is selected based on successive KPSS unit-root tests are considered as models. The procedure successively tests higher order differences of the data for a unit root until a non-significant p value is observed.

There are too many parameter combinations of p, q, and d to afford an exhaustive search for the model with global best AIC. Therefore, a step-wise algorithm is applied. First, the four models ARIMA$(2, d, 2)$, ARIMA$(0, d, 0)$, ARIMA$(1, d, 0)$, and ARIMA$(0, d, 1)$ are fitted with $c \neq 0$ if $d \leq 1$ or with $c = 0$ otherwise. The model with the best AIC is designated as the *current* model. Second, variations of the current model are considered by varying the model parameters by ± 1 in an iterative process, respecting several constraints on the fitted models. When a model of better AIC is discovered, it becomes the new current model, until no variation of the current model with lower AIC is found. The final current model is used to produce forecasts. [1]

Automated ETS *Models.* As shown in [1], exponential smoothing methods are equivalent to optimal forecasts from innovations state space models. We thus consider the class of all innovations state space models as the pool for model selection in automated ETS modeling. To distinguish model structures, the notation ETS(error, trend, seasonality) is employed. The error component can be either additive or multiplicative, the trend component can be either missing, additive, additive damped, multiplicative, or multiplicative damped, and the seasonality component can be either missing, additive, or multiplicative. Considering all combinations, there are 30 model structures. In our case, as our input data is not strictly positive and already seasonally adjusted, multiplicative error models are not applicable, and the seasonality component may be disabled (missing), reducing the pool to only 5 model structures.

All 30 (in our case only 5) innovations state space model structures share a general layout consisting of a state vector $\boldsymbol{x}_t := (l_t, b_t, s_t, s_{t-1}, \ldots, s_{t-m+1})'$ and the state space equations

$$y_t = w(\boldsymbol{x}_{t-1}) + r(\boldsymbol{x}_{t-1})\epsilon_t \tag{6}$$

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}) + g(\boldsymbol{x}_{t-1})\epsilon_t. \tag{7}$$

In these equations, $\{\epsilon_t\}$ is a Gaussian white noise process with mean zero and variance σ^2, and $\mu_t = w(x_{t-1})$. In the model with additive errors, $r(x_{t-1}) = 1$ holds, so that $y_t = \mu_t + \epsilon_t$. To calculate point forecasts until horizon h, these equations can be iteratively applied for $t = n+1, n+2, \ldots, n+h$, while setting $\epsilon_{n+j} = 0$ for $j = 1, \ldots, h$.

Parameters for these innovations state space models are obtained by maximum likelihood estimation. The model structure most appropriate for the input data at hand can then be selected based on AIC, leading to the automated ETS forecasting algorithm of [1]:

1. Apply all model structures to forecast the input time series, choosing model parameters by maximum likelihood estimation.
2. Select the model with the best AIC.
3. Use this model to produce h point forecasts.

4.3 DataModeler

In addition to GreenPocket's approach and our ensemble approach a state of the art modeling approach based on Genetic Programming (GP), namely Evolved Analytics' DataModeler, will be included in our study as the third modeling tool. [5]

We would argue that the challenge of predicting one month of energy consumption based on three months worth of data is not a conventional problem for symbolic regression (SR) modeling with GP. Symbolic regression is a methodology for finding global relationships in data and for global approximation. SR via GP uses a stochastic iterative search process (evolution) to evolve appropriate model forms using supplied sets of input variables, functional primitives, and constants. Models are developed to achieve optimal trade-offs between prediction accuracy on provided input-response data and model complexity as well as numerical stability. SR is particularly useful for finding laconic expressions for smooth albeit high-dimensional response surfaces.

The main goal of applying ensemble-based symbolic regression to the non-smooth data was to see whether this flexible methodology is capable to appropriately identifying the time lags and combine them together into acceptable dynamic models. We used ensemble-based symbolic regression implemented in DataModeler [2], because it is best suited for modeling very high dimensional data with a only small fraction of input variables significantly related to the response. Due to space limitations we only report the results of GP experiments which used variations of lagged consumption as input variables. While predicting energy consumption one day ahead did not pose real challenges for GP, achieving success for predictions one month ahead has hit a wall of decreased prediction accuracy on both training and test data. To predict four weeks ahead we were forced to set the considered time-lags to 28 days (2688 intervals) and earlier.

In described experiments we used the quarter-hourly time lags between 28 and 35 days from the moment of prediction (672 time-lags, i.e. input variables), and we also constructed a reduced set of inputs of all quarter-hour intervals between 28 and 29 days ago, lags between exactly 28 and 35 days ago, and lags between 28 days and one quarter-hours and 35 days and one quarter-hour ago – 110 variables in total. The variables used in the experiment are:

d_{2688}–$d_{2784}, d_{2880}, d_{2976}, d_{3072}, d_{3168}, d_{3264}, d_{3360}, d_{2785}, d_{2882}, d_{2979}, d_{3076}, d_{3173}, d_{3270}, d_{3367}$, where $d_i(t) = c(t-i)$, for $c(t)$ being a quarter-hourly energy consumption at time moment t. Because of the large backshift we could only use 5093 records from the given training data (63%). All GP runs used arithmetic operators and *square*, *minus*, *inverse* as functional primitives, random constants sampled from the interval $[-5, 5]$, and time-limited evolutions of 20 minutes each (for other default GP settings see [2]). In the first evolutions a consistent set of driving variables was discovered (see Figure 2). The top ten driving variables were further used for new runs in a reduced space with all other settings unchanged.

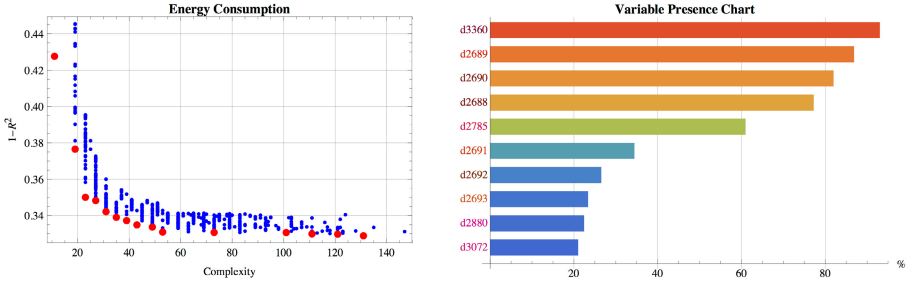


Fig. 2. GP results of the first experiments: Selected Model Set and Driving Variables. Left plot presents the selected set of models plotted in the space of model complexity (sum of nodes of all subtrees corresponding to the parse-tree defining the model) and model accuracy (1-R^2). Variable presence in the right plot stands for the fraction of models in the selected set containing variable in question.

The runs in the second batch generated models of higher prediction accuracy and smaller complexity. From these models we selected an ensemble of models using the SelectModelEnsemble. DataModeler defines final model prediction at a certain point as an average of five predictions closest to the median of predictions of ensemble models at this point. The model closest to the median prediction on test data was the model from the knee of the Pareto Front of the ensemble models:

$$t(i) = 8.94 - \frac{253.62}{28.15 + t(i-2688) + t(i-2689) + t(i-2690) + t(i-2785) + t(i-3360)}.$$

Even without constant optimization the model alone produces the prediction error on the test set of RMSE $= 1.03$. This accuracy is comparable with an accuracy of the RMSE $= 0.97$ of the golden batch week prediction constructed by averaging available consumption per day of the week (from Monday to Sunday) and predicting test data only using day of the week and a quarter-hour time moments.

Our results indicate that ensemble-based symbolic regression while used for constructing global approximation of high-dimensional data is capable of identifying appropriate time lags and creating small and very interpretable dynamic models in such a challenging problem like energy consumption.

5 Results and Discussion

The predicted energy consumption for all four models is shown in Figure 3. From that figure it is easy to see that the GreenPocket prediction for each day is the same which is explained by the fact that their model has no covariates and only carries the average daily consumption profile of the last 14 days of the training data forward. Consequently, it fails to predict the days of low consumption (Sundays) in the test data. Both the ETS as well as the ARIMA prediction are dominated by the seasonal effect which is estimated by the STL procedure. Therefore their predictions differ only slightly. If we check the chosen ARIMA model, we see that it has, as expected, no lag. The ETS model is a simple additive model without any periodicity. The prediction from DataModeler, the GP system in our comparison, appears quite different. It has a lower intraday variance and mispredicts the consumption on "off" days by a small but noticeable offset.

It is unclear which of the three models is the "best". In practice there are likely no measurable differences between the ARIMA and ETS model so one would choose the model with the lower computational burden. This is confirmed by the three error measures depicted in Table 1. The results of the ensemble-based approaches pass a preliminary test of generalizability, as the result ranking of `meter1_series2` matches the result ranking of `meter1_series1`.

According to those error measures, the predictions by DataModeler are always slightly better, regardless of the error measure used. This is not all that surprising given that DataModeler can choose its model from a much broader and richer class of models.

Overall it is entirely not clear which error measure is the most appropriate for this problem. All of the models discussed still have weaknesses which are not necessarily captured by the error measures used. In the future we want to investigate loss functions which model the actual business case—Euros saved or spent based on the

(mis)prediction of the model. This would require yet another time series, the energy prices per quarter hour, and its prediction to evaluate a joint model for a purchasing strategy.

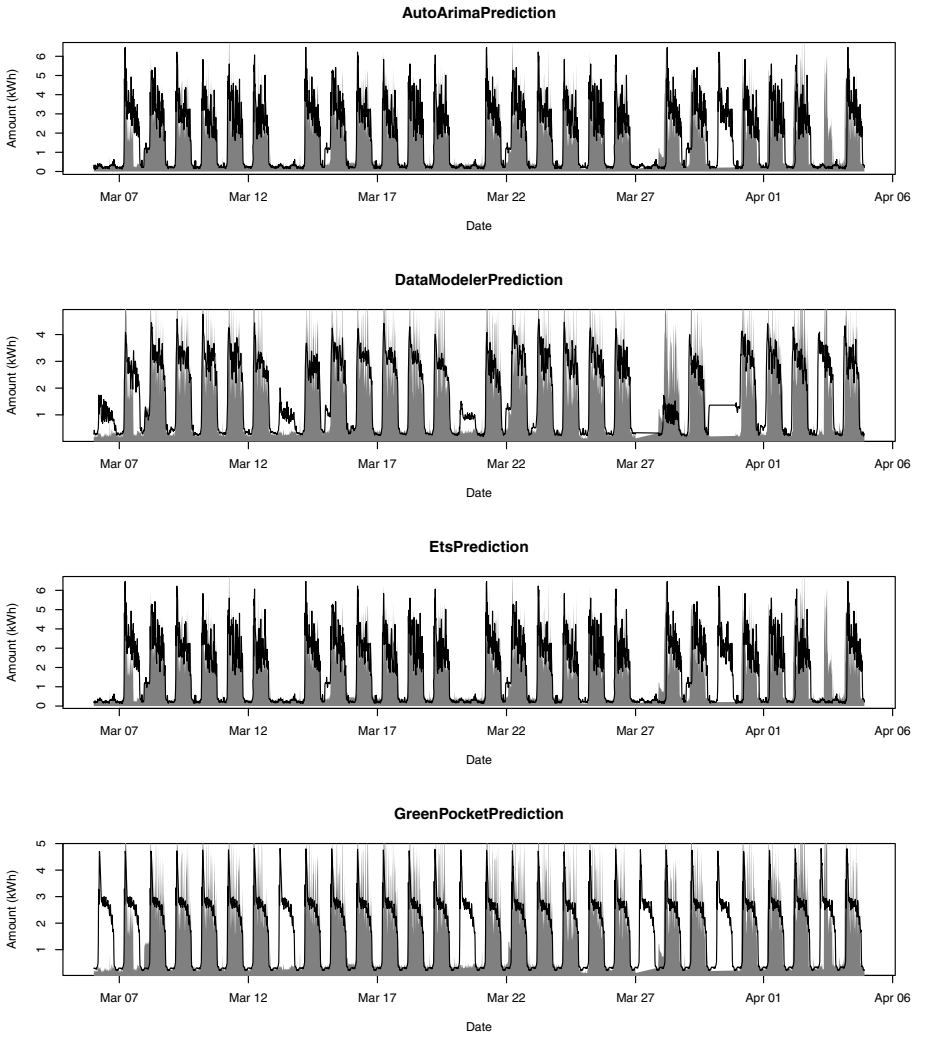


Fig. 3. Time series plots of the test data range predictions generated by the methods studied in this paper. From top to bottom: Ensemble-based ARIMA (AutoArimaPrediction), symbolic regression (DataModelerPrediction), ensemble-based exponential smoothing (EtsPrediction), GreenPocket baseline (GreenPocketPrediction). The true energy consumption time series is shown as a gray backdrop in each plot.

Table 1. Experiment results for time series 1 and 2. Error measure values were truncated to 3 decimal digits. Best error values are shown in bold font. DataModeler results for time series 2 where not available at the time of publication.

	Series 1			Series 2		
Method	RMSE	MAE	RMSElog	RMSE	MAE	RMSElog
Automated ARIMA	1.157	0.710	0.352	**0.962**	**0.596**	**0.235**
Automated ETS	1.157	0.710	0.352	0.967	0.605	0.239
DataModeler	**1.030**	**0.699**	**0.328**			
GreenPocket	1.151	0.741	0.382	1.196	0.763	0.360

6 Conclusions and Outlook

The two ensemble-based time-series prediction methods are easily applicable and are able to provide precise forecasts. Regarding research question **Q1** posed in Section 2 of this work, our experiments show that modern ensemble-based approaches, without further domain knowledge about the data used, are able to compete with custom-built approaches in our real-world energy consumption time-series prediction scenario.

GP and symbolic regression in particular is a promising model generation strategy, because it can find structure as well as driving variables at the same time. Given such a model, the constants can then be adapted to fit different data from the same domain. It should also be applicable to other consumption problems. Regarding research question **Q2**, our results show that symbolic regression, as a generic method, is able to create time-series prediction models that are as accurate as custom-built approaches, at least in our application scenario. All models studied in this work are efficiently executable, making them applicable to the large data volumes common in smart metering.

In future work we will investigate whether the models constructed for one particular customer can be used to predict energy consumption of other customers by merely re-fitting model parameters.

References

1. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting the forecast package for r. Journal of Statistical Software 27(3), 1–22 (2008)
2. Evolved Analytics LLC: DataModeler Release 8.0 Documentation. Evolved Analytics LLC (2011)
3. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: STL: A Seasonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics 6(1), 3–33 (1990)
4. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2012) ISBN: 3-900051-07-0
5. Kordon, A.K., Smits, G., Kotanchek, M.E.: Industrial evolutionary computing. In: Thierens, D. (ed.) GECCO (Companion), pp. 3297–3322. ACM (2007)

Evolving Non-Intrusive Load Monitoring

Dominik Egarter[1], Anita Sobe[2], and Wilfried Elmenreich[1,3]

[1] Institute of Networked and Embedded Systems / Lakeside Labs
Alpen-Adria-Universität Klagenfurt, Austria
[2] Institut d'informatique, Université de Neuchâtel, Switzerland
[3] Complex Systems Engineering, Universität Passau, Germany
{dominik.egarter,wilfried.elmenreich}@aau.at, anita.sobe@unine.ch

Abstract. Non-intrusive load monitoring (NILM) identifies used appliances in a total power load according to their individual load characteristics. In this paper we propose an evolutionary optimization algorithm to identify appliances, which are modeled as on/off appliances. We evaluate our proposed evolutionary optimization by simulation with Matlab, where we use a random total load and randomly generated power profiles to make a statement of the applicability of the evolutionary algorithm as optimization technique for NILM. Our results shows that the evolutionary approach is feasible to be used in NILM systems and can reach satisfying detection probabilities.

Keywords: Evolutionary Algorithm, Knapsack Problem, Evolution, Non-Intrusive Load Monitoring, NILM.

1 Introduction

With the upcoming of decentralized regenerative energy sources, the amount of available energy at a particular time and, due to network capacity constraints, location becomes dependent on the current weather situation (photovoltaic production depends on amount of sunshine, windmill-powered plants on wind speed). One way to mitigate this issue is to provide energy storage (e. g., by batteries, pumped-storage hydropower plants, conversion to methane, etc). The other way is shaping the energy consumption at the consumer side. A typical household contains hundreds of electric appliances, whereof a few dozen are relevant in terms of energy consumption. In order to keep the convenience level for the customer high, we need an intelligent control system that identifies devices currently turned on and proposes minimal-invasive changes to their usage. To get this information, each relevant appliance could be equipped with a smart meter or an embedded communication and control interface able to deliver power information and characteristics [5]. Upgrading all devices in a current household this way would be painstaking and costly. An alternative approach is non-intrusive load monitoring (NILM)[8], which determines and classifies individual appliances based on characteristic load profiles. For identification only a *single* smart meter measuring the total power consumption with appropriate timely resolution

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 182–191, 2013.
© Springer-Verlag Berlin Heidelberg 2013

is sufficient. NILM extracts features like active power, frequency etc., classifies appliances and identifies appliances by matching the measured data to a reference database. Thus, the identification can be described as an optimization problem of superimposed power profiles. Possible solutions for this problem are optimization techniques like integer linear programming [16] or pattern recognition methods like artificial neural networks [4]. In recent years, the technique of NILM has been extended and improved, but up to now no universal solution has been developed [18].

We propose an evolutionary optimization approach that identifies a variable number of appliances by their given power profile. The idea is that the potential appliance profiles (out of a database) have to be matched with the given power profile with minimum error [2]. The presented problem is related to the Knapsack problem, which is NP-hard [6, 14]. Possible techniques to tackle the Knapsack problem are either exact, heuristic or meta-heuristic solutions [11]. Genetic algorithms have successfully been used for handling the Knapsack problem during the last twenty years. Implementations are ranging from solving the simple 0-1 knapsack problem [15] to more lavish techniques like hybrid optimization [17] and multidimensional Knapsack problems techniques [9].

In the context of NILM, the genetic algorithm is typically used for detecting features and patterns of appliance power profiles [1, 3] and for optimizing existing parameters which are used in fuzzy systems [13]. Furthermore, Leung, Ng and Cheng presented in [12] a possible approach to use the genetic algorithm to identify appliances. In their paper they grouped power signatures out of one load signature data set into the groups sinusoid, quasi-sinusoid and non-sinusoid load signatures by averaging 50 consecutive one-cycle steady state current waveforms. They built a composition of load signatures of the same group between each other and a composition of load signatures among the groups. Finally, they used the genetic algorithm to identify the wanted load signatures. In contrast to this approach, we use the entire power load signal of a household over two hours and not the mean current waveform of 50 consecutive one-cycles. Further, we do not split up the appliances into groups. We randomly generate common power profiles in steady state and detect these power profiles in a random superimposed composition of power profiles over a time window of two hours. Accordingly, we make a statement about which appliance have been used and also at which point in time. The remainder of this paper is organized as follows: in Section 2 we describe the optimization problem of overlapping power profiles in more detail and how it can be solved with the help of the evolutionary algorithm. In Section 3 we evaluate the presented genetic algorithm by different test scenarios like algorithm dependence on the number of wanted power profiles or the detection behavior under the influence of noise. Finally, in Section 4 we conclude this paper and present future work.

2 Evolutionary Appliance Detection

The knapsack problem is a well-known optimization problem with the aim of packing a set of n items with a certain weight w_i and profit d_i into a knapsack

of capacity C in the most profitable way. If it is possible to place a item into the knapsack without exceeding the capacity C by using $x_i \in \{0, 1\}$, which is responsible for whether or not a certain item is used, a profit d_i is earned. This context can be summarized as follows:

$$\text{maximize} \sum_{i=1}^{n} d_i \cdot x_i \tag{1}$$

$$\text{subject to} \sum_{i=1}^{n} w_i \cdot x_i \leq C. \tag{2}$$

The problem of packing items into a desired shape can easily be compared to the appliance detection and classification in NILM systems. NILM has the major aim of detecting and identifying appliances according to their own power profile P_i in the measured total load $P(t)$. The power profiles P_i are characterized by their power magnitude m_i and time duration τ_i and the total load is given by:

$$P(t) = \sum_{i=1}^{n} P_i \cdot a_i(t) + e(t), \tag{3}$$

where n is the number of known and used appliances, $a_i(t) \in [0, 1]$ represents the state timing vector of the appliance being on ($a_i(ts) = 1$) at switching time t_s or off ($a_i(t) = 0$) and $e(t)$ describes an error term. Therefore the general optimization problem of NILM can be formulated as the minimum error $e(t)$ of the total power load and the composite appliance power profiles P_i:

$$e(t) = \arg\min \left| P(t) - \sum_{i=1}^{n} P_i \cdot a_i(t) \right|. \tag{4}$$

In contrast to the traditional knapsack problem, which allows only bounded values smaller than capacity C, we allow positive and negative error values and take the absolute error value for our fitness evaluation. Turning back, a NILM system tries to find the right switching points $a_i(t)$ and their corresponding appliances to minimize the error between the sum of superimposed appliance power profiles and the total load $P(t)$. This relates to the knapsack problem, where in the case of NILM the capacity C of the knapsack corresponds to the total load $P(t)$ and the items of the knapsack correspond to the appliance power profiles P_i. Further, we assume that the profit d_i equals 1, because we suppose that all appliances in the household are of equal importance concerning their usage. The aim of the evolutionary approach is to find a composition of power profiles P_i, which can be packed into the measured total load $P(t)$ with minimum error. Therefore, we modify the general knapsack problem by dismissing the profit maximization with an error minimization. An illustration of the basic principle can be seen in Figure 1, where a collection of possible power profiles P_i and the trend of the total power load are presented. Out of the collection a selection of power profiles P_i is met and this selection is then packed into the trend of the total power load to best approximate the trend of the total power load.

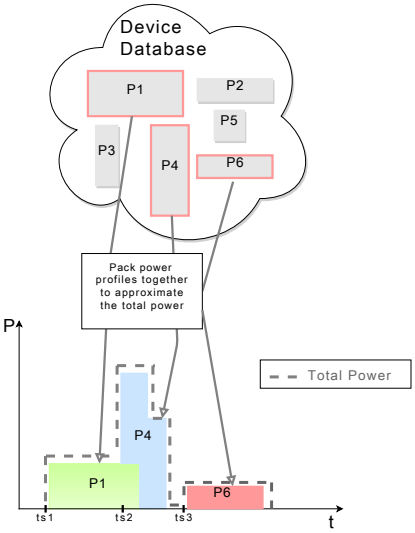


Fig. 1. Basic principle of the ON/OFF time genome appliance detection. Find a composition of saved power profiles P_i, place it at the switching times t_s and try to minimize the error between the evolved power load and the given total power load.

To solve this error minimization, we use an evolutionary algorithm as described in [7] with uniform mutation, single point crossover and elite selection. In detail, the used evolutionary algorithm has to evolve the set of used power profiles P_i. To be able to reduce the complexity of the optimization problem we assume that we know the starting points of the appliances' power profiles. A possible technique to detect the switching times t_s is mentioned by Hart in [8] and is called edge detection of $P(t)$. The edge detection calculates the difference of the current power signal $P(t)$ and the delayed power signal $P(t-1)$ and tries to detect the switching event by thresholding the calculated difference value. A switching time t_s is given if the difference value is larger than a predefined threshold d and accordingly, $a_i(t_s)$ is set to 1.

Thus, the evolutionary algorithm examines a composition of appliance power profiles P_i, place them at the switching time t_s by multiplying P_i with its corresponding state timing vector $a_i(t)$ and approximate the total load $P(t)$. Therefore, a genome maps a set of power profiles P_{xi}, where x_i represents the index of the power profile[1] P_i stored in the database. The fitness function F_s for the optimization is given by:

$$F_s = -\left| P(t) - \sum_{i=1}^{Nb} P_{x_i} \cdot a_i(t) \right|, \tag{5}$$

[1] We assume that each power load profile P_i is only stored once in the database. The database has a size of db.

where Nb represents the number of switching times t_s and correspondingly, also the number of used appliances, because we assume that every appliance occurs only ones at different switching times t_s. According to presented fitness function F_s, the best achievable fitness value is $F_s = 0$, which corresponds to an error of 0 between the evolved power load and the total power load $P(t)$.
In the following section we evaluate the detection ability of the presented modified knapsack problem by the evolutionary algorithm.

3 Evaluation

To be able to evaluate the presented evolutionary algorithm with its genome representation and fitness function, we compute simulations of the evolutionary algorithm in Matlab. The parameter properties for the evolutionary algorithm can be found in Table 1 and were determined empirically. We performed S simulation runs for each test case and generated the mean fitness $F = \sum_{s \in S} \frac{F_s}{S}$ and the mean detection probability $\bar{P}_{det}$. The mean detection probability $\bar{P}_{det}$ is given by $\bar{P}_{det} = \sum_{s \in S} P_{det}/S$, where P_{det} is given by $P_{det} = \frac{\#det}{Nb}$ and is the detection probability by simulation run. The variable $\#det$ is the counted

Table 1. Parameters used for the evolutionary algorithm and the simulations

Variable	Description	Value range
P_{elite}	Elite selection	10%
P_{mutate}	Uniform mutation	40%
$P_{crossover}$	Single point crossover	40%
P_{new}	New individuals	10%
$P_{mutateRate}$	Mutation rate	10%
G	Number of generations for the GA	500
N	Number of populations for the GA	500
P_i	Power profile P_i of appliances	
m_i	Randomly generated power magnitude[2] m_i	$m_i \in [100, 4000]$
τ_i	Randomly generated time duration[3] τ_i	$\tau_i \in [60, 3600]$
P_g	Random generated total power load over two hours in seconds resolution ($T = [1, 7200]$). Total power load equals a random set of power profiles P_i out of the database.	
db	Number of random generated and stored power profiles P_i	50
Nb	Number of used appliances	5
S	Simulation runs	10

[2] The power magnitude m_i was chosen in a common power range of household appliances.
[3] The time duration τ_i represents a common usage of household appliances between one minute and one hour. The time window of 2 hours will produce a variety of superimposed power loads.

number of correctly detected power profiles P_i and Nb is, as mentioned before, the number of switching events and also the number of used power profiles[4] P_i. Beyond that, Table 2 show how many errors $e_s = N_b - \#det$ occurred for every simulation run $s \in S$ and further, we calculate the mean error $\bar{e} = \sum_{s \in S} e_s / S$ per the simulation runs $s \in S$. For the evaluation we used the following different test scenarios:

- influence of the number of wanted and stored power profiles and
- influence of disturbances like noise and unknown power loads

to make a statement on their influence on the detection ability of the presented NILM technique, which we will describe in the following sections in more detail.

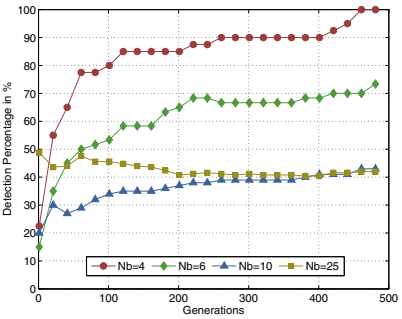
3.1 Variation of Wanted and Stored Appliances

In Figure 2 the results for different numbers of active devices and different sizes of the power profile database is shown. First, we consider the case of varying number of power profiles. For this test scenario we chose a database size $db = 50$ and the number of wanted power profiles P_i was $Nb = [4, 6, 10, 25]$. We have taken these values of Nb to cover the cases of household common and high Nb appropriate to the size of db and the considered period of time (2 hours). Considering Figure 2, the fitness F and detection percentage Θ reach satisfying and sufficient results of a detection percentage up to 100%. We can see that the fitness value F of Figure 2(c)[5] depends on the number of devices (curves are from low Nb at top of the figure to high Nb at the bottom). The lower the number, the better the fitness, because it is harder to find a set of correctly ordered power loads of size 10 than of size 5. The evolutionary algorithm is able to find sufficient results after 100 to 200 generations, which is acceptable for the intended application scenario. The detection behavior is similar in the case of the detection percentage Θ in Figure 2(a). The lower the number of used power loads, the better is the result of detected power loads. We claim that the worst case in this example is finding 25 power loads, reasoning that our optimization problem can be seen as a problem of combining several things k out of a larger group n, where the order is not taken into consideration and accordingly, can be considered as the well known combination problem. A combination can be formulated as $\frac{n!}{k! \cdot (n-k)!}$ and the worst case for this problem is, if $k = n/2$, which is in our case with $k = Nb = 25$ and $n = db = 50$.

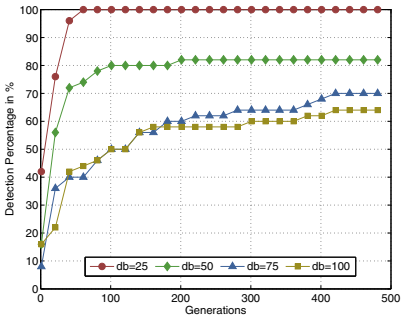
Beyond that, we varied the number of stored power profiles db in the second test scenario in Figure 2(b). For this we used a database size of $db = [25, 50, 75, 100]$ and $Nb = 5$ to make a statement regarding scalability of the

[4] Every appliance power load P_i can only be used once and accordingly, the number of wanted power profiles is the same like the number of switching times t_s.

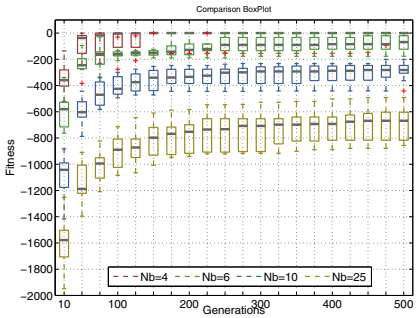
[5] The boxplot of the fitness trend for Nb variation should give a general impression how the fitness is evolving over generations and is comparable for fitness evaluation and detection probability, because the fitness has a direct relation to the detection probability.



(a) Detection percentage Θ for Nb Variation



(b) Detection percentage Θ for db Variation



(c) Boxplot of fitness F for Nb Variation

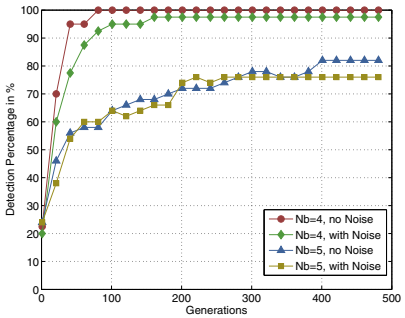
Fig. 2. This figure shows the trend for the varying $Nb = [4, 6, 10, 25]$, for varying $db = [25, 50, 75, 100]$ with $Nb = 5$ and also the boxplot of the mean fitness F for varying Nb

evolutionary algorithm. We can see that the evolutionary algorithm is able to reach a high detection percentage Θ dependent on used db. If the number of db is increased, the evolutionary algorithm evolves a lower Θ, because the search space is becoming bigger.

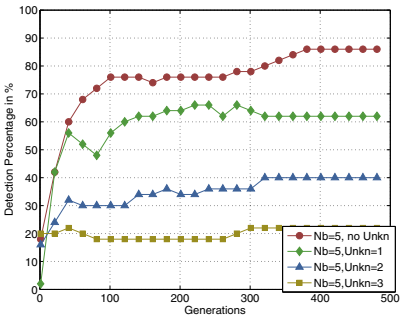
In addition, we present the mean error $\bar{e}$ and the mean detection probability $\bar{P}_{det}$ of the detection process in Table 2. According to this table, we can claim that the lower Nb, the better the result of no errors and that the evolutionary algorithm is dependent on db. Finally, the simulations shows, that the detection depends on the characteristic of overlapping power loads. The more power loads are superimposed, the more difficult it is to make a correct decision and to minimize the error between the total power load and the evolved power load. In our test scenarios the probability of overlapping power loads is rather high, because of the chosen time duration of $\tau_i = [60, 3600]$ in a time window of 2 hours ($T = [1, 7200]$) and correspondingly, the detection algorithm still works sufficiently and satisfying.

Table 2. Table for error e_s by simulation run S and the mean error $\bar{e}$ for varying $Nb = [6, 10, 25, 45]$ with $db = 50$, varying $db = 100$ with $Nb = 5$, under the influence of noise and under the influence of unknown power loads $Unkn = [1, 2]$

	Detection errror e_s and mean detection probability $\bar{P}_{det}$ by								
	Nb-Variation					db-Variation[6]	with Noise[7]	with unknown[6]	
	4	5	6	10	25	100		1	2
$\bar{e}$	0	0.7	1.6	5.7	14.6	1.8	0.9	1.9	3
$\bar{P}_{det}$ in %	100	86	73	43	41	64	82	62	40



(a) Trend of the detection percentage Θ with noise



(b) Trend of the detection percentage Θ with unknown power loads

Fig. 3. This figure shows the trend under the influence of noise and unknown power loads $Unkn = [1, 2, 3]$. The size of $db = 50$ and $Nb = [4, 5]$.

3.2 Influence of Disturbances

To make a better statement of the ability and the quality of the presented algorithm, we examined the detection behavior of the evolutionary algorithm under the influence of noise and unknown, not stored appliances. In Figure 3(a) noise with zero mean $\mu = 0$ and the standard deviation $\sigma = \sqrt{max(P(t))}$ was added to our simulated total power load $P(t)$ and we added unknown power loads $Unkn = [1, 2, 3]$ to the total load $P(t)$ in Figure 3(b). At first, we consider the detection scenario under the influence of noise. In this scenario, the simulation results show that the detection percentage Θ in Figure 3(a) is slightly influenced by noise and therefore, we claim that the presented algorithm is robust to noise. We observe this behavior in the cases of $Nb = 4$ and $Nb = 5$. Both test scenarios show satisfying detection results of 0 errors. Further, we consider the case of adding unknown appliances to the total load $P(t)$ in Figure 3(b). This figure indicates that the detection percentage Θ depends on the number of unknown power loads[8], because the more unknown information is in the system the more

[6] $Nb = 5$.
[7] $Nb = 5$, $\mu = 0$,$\sigma = \sqrt{max(P(t))}$.
[8] Nb=5 for this test scenario.

complicated it is for the evolutionary algorithm to establish a correct evolved composition of power loads P_i to approximate $P(t)$. Finally, we also present the mean error $\bar{e}$ and the mean detection probability $\bar{P}_{det}$ in Table 2, which confirms our statements that our algorithm is noise robust and is dependent on the number of known and accordingly, on the number of unknown appliances.

4 Conclusion and Future Work

In this paper we present an evolutionary algorithm to solve the task of detecting an appliance based on their power loads in the total load of a household over a time window of 2 hours. Our algorithm provides promising results to detect superimposed respective power loads with up to 100% certainity. In more detail, the presented algorithm has the following detection characteristics:

- Detection percentage Θ up to 100% depending on Nb and db
- The higher Nb and the higher db, the lower the detection percentage Θ
- Sufficient results at generation $G > 100$
- Robustness against noise
- Dependent on additive unknown and the quantity of overlapping power loads

Our results show that the presented algorithm is feasible for use in NILM systems and can achieve a detection probability of 100% in case of low number of devices Nb and records used power loads even if not all power loads are detected correctly. With our algorithm applied to a real household the results can be used for tracking the used power consumption and the usage of appliances and can improve the energy-awarness concerning the energy consumption of appliances. The current version of the algorithm was tested for on/off appliances with constant time duration. In future work, we plan to extend our algorithm for arbitrary shapes of power profiles and to evaluate the approach using real appliances [10].

Acknowledgments. This work was supported by Lakeside Labs GmbH, Klagenfurt, Austria and funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant KWF-20214 | 22935 | 24445. We would like to thank Kornelia Lienbacher for proofreading the paper.

References

[1] Baranski, M., Voss, J.: Genetic algorithm for pattern detection in nialm systems. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 3462–3468 (2004)

[2] Bijker, A., Xia, X., Zhang, J.: Active power residential non-intrusive appliance load monitoring system. In: AFRICON 2009, pp. 1–6 (September 2009)

[3] Chang, H.H., Chien, P.C., Lin, L.S., Chen, N.: Feature extraction of non-intrusive load-monitoring system using genetic algorithm in smart meters. In: IEEE 8th International Conference on e-Business Engineering (ICEBE), pp. 299–304 (2011)
[4] Chang, H.H., Lin, C.L., Lee, J.K.: Load identification in nonintrusive load monitoring using steady-state and turn-on transient energy algorithms. In: 14th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 27–32 (2010)
[5] Elmenreich, W., Egarter, D.: Design guidelines for smart appliances. In: Proc. 10th International Workshop on Intelligent Solutions in Embedded Systems (WISES 2012), pp. 76–82 (2012)
[6] Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
[7] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
[8] Hart, G.: Nonintrusive appliance load monitoring. Proceedings of the IEEE 80(12), 1870–1891 (1992)
[9] Hoff, A., Løkketangen, A., Mittet, I.: Genetic Algorithms for 0/1 Multidimensional Knapsack Problems. In: Proceedings Norsk Informatikk Konferanse, NIK 1966 (1996)
[10] Kolter, J.Z., Johnson, M.J.: REDD: A Public Data Set for Energy Disaggregation Research. In: Proceedings of the SustKDD Workshop on Data Mining Applications in Sustainability (2011)
[11] Lagoudakis, M.G.: The 0-1 Knapsack Problem An Introductory Survey. Technical report
[12] Leung, S.K.J., Ng, S.H.K., Cheng, W.M.J.: Identifying Appliances Using Load Signatures and Genetic Algorithms. In: International Conference on Electrical Engineering, ICEE (2007)
[13] Lin, Y.H., Tsai, M.S., Chen, C.S.: Applications of fuzzy classification with fuzzy c-means clustering and optimization strategies for load identification in nilm systems. In: 2011 IEEE International Conference on Fuzzy Systems (FUZZ), pp. 859–866 (2011)
[14] Martello, S., Toth, P.: Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc., New York (1990)
[15] Singh, R.P.: Solving 0/1 Knapsack problem using Genetic Algorithms. In: IEEE 3rd International Conference on Communication Software and Networks (ICCSN), pp. 591–595. IEEE (2011)
[16] Suzuki, K., Inagaki, S., Suzuki, T., Nakamura, H., Ito, K.: Nonintrusive appliance load monitoring based on integer programming. In: SICE Annual Conference, pp. 2742–2747 (2008)
[17] Cotta, C., Troya, J.: A hybrid genetic algorithm for the 0-1 multiple knapsack problem. Artificial Neural Nets and Genetic Algorithms 3, 251–255 (1998)
[18] Zeifman, M., Roth, K.: Nonintrusive appliance load monitoring: Review and outlook. IEEE Transactions on Consumer Electronics 57(1), 76–84 (2011)

On the Utility of Trading Criteria Based Retraining in Forex Markets

Alexander Loginov and Malcolm I. Heywood

Faculty of Computer Science,
Dalhousie University, Halifax, NS, Canada
a.loginov@yahoo.ca, mheywood@cs.dal.ca
http://www.cs.dal.ca

Abstract. This research investigates the ability of genetic programming (GP) to build profitable trading strategies for the Foreign Exchange Market (FX) of three major currency pairs (EURUSD, USDCHF and EURCHF) using one hour prices from 2008 to 2011. We recognize that such environments are likely to be non-stationary. Thus, we do not require a single training partition to capture all likely future behaviours. We address this by detecting poor trading behaviours and use this to trigger retraining. In addition the task of evolving good technical indicators (TI) and the rules for deploying trading actions is explicitly separated. Thus, separate GP populations are used to coevolve TI and trading behaviours under a mutualistic symbiotic association. The results of 100 simulations demonstrate that an adaptive retraining algorithm significantly outperforms a single-strategy approach (population evolved once) and generates profitable solutions with a high probability.

Keywords: Coevolution, non-stationary, FX, Forex, Currency.

1 Introduction

The Foreign Exchange (FX) Market is the world biggest financial market which produces 1/3 of all financial transactions in the world [1]. The average daily turnover of FX was almost $4 trillion in 2010 and it was 20% higher in April 2010 than in April 2007 [2]. An FX market consists of currency pairs which are weighted by economic conditions for that specific denomination versus any or all others in the marketplace. Thus, the perceived value of a currency is a reflection of the market's ranking for that denomination's economy on any given day. An FX market is technically 'purer' than a stock market i.e., a currency price action reacts more strongly to resistance and support levels than equity market does [3]. All the above factors make FX markets very attractive for traders and expands the demand for automated trading systems, albeit under demanding conditions.

However, the underlying data describing such trading environments is typically non-stationary. Thus, assuming the classical approach of training over fixed partitions of data (e.g., training, validation and test) results in brittle solutions that could be specific to the partition on which they are evolved [4], Chapter 7. Moreover, the use of validation data in financial data forecasting is not in itself a "silver

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 192–202, 2013.
© Springer-Verlag Berlin Heidelberg 2013

bullet" [5]. One solution proposed to this problem is to train on a continuous basis, while explicitly maintaining population diversity [4], Chapter 8. Thus, following the initial evolution of a population of trading agents, taking G generations, the best solution trades for 'n' days. The same 'n' trading days are then employed to readapt the population for $g < G$ generations. However, the selection of appropriate g and n is in itself a function of the data (market dynamic). Another approach might be to coevolve a subset of the training partition [6] or to combine coevolution of the training partition with an explicitly streaming context [7].

In this work we investigate a somewhat different approach. As emphasized by the study of [4], we recognize that under non-stationary environments assuming a modular representation can be beneficial. There are several ways of potentially achieving this. In this work we adopt a symbiotic approach to genetic programming (e.g., [8,9]). Specifically, trading indicators (TI) are coevolved with trading decision trees (DT). Assuming a symbiotic approach enables us to adopt unique representations for each while searching both representations in a co-ordinated way. Secondly, we define specific trading criteria that characterize when retraining should take place. Thus, relative to a fixed training–validation parameterization, training is only ever re-triggered by the trading criteria. Now, *relative to the point of re-triggering*, an entirely new training–validation sample is defined to evolve a new trading agent. Naturally, there is a trade off with regards to the cost of retraining versus maintaining a population of previously evolved models for redeployment e.g., [10]. Under this particular deployment, the retraining cost is significantly smaller than the interval between consecutive data pairs, thus the retraining cost does not detract from 'real-time' operation.

The proposed approach – hereafter FXGP – is demonstrated relative to the task of FX trading under three currencies for 3 year periods in each case. A base case employing the same symbiotic representation but without retraining provides a performance baseline. Conversely, adding the capability to dynamically identify retraining points through trading criteria results in a significant improvement.

1.1 Glossary

Hereafter the following terminology will be assumed [11]:

Ask: Price at which broker/dealer is willing to sell. Also referred to as "Offer".

Balance: The value of your account not including unrealized gains or losses on open positions.

Bid: Price at which broker/dealer is willing to buy.

Drawdown: The magnitude of a decline in account value, either in percentage or currency terms, as measured from peak to subsequent trough.

Fundamental Analysis: Macro or strategic assessment of where a currency should be trading on any criteria but the price action itself. The criteria often includes the economic condition of the country that the currency represents e.g., monetary policy, and other "fundamental" elements.

Leverage: The amount, expressed as a multiple, by which the notional amount a trader exceeds the margin required to trade. The specific approach taken to this is described in Section 3.

Pip: The smallest price increment in a currency. Often referred to as "ticks" in the future markets. For example, in EURUSD, a move of 0.0001 is one pip.

Spread: The distance, usually in pips, between the Bid and Ask prices. Section 3 details the fixed spread assumed in this work.

Stop: Also called "Stop Loss" or "S/L". An order to buy or sell when the market moves to a specific price.

Technical Analysis: Analysis applied to the (price) action of the market to develop a trading decision, irrespective of fundamental factors.

2 Proposed Algorithm

2.1 The FXGP Algorithm Overview

As established in the introduction, a cycle of evolution is initially completed against the first 1,000 records (N_t) and validated against the next 500 (N_v); or ≈ 2 months and 1 month respectively. Trading then commences until some failure criteria is satisfied – e.g., an excessive drawdown is encountered – at which point the $N_t + N_v$ records leading up to the failure are used to evolve a new trading agent, and the process repeats. Note, however, that such a scheme is distinct from the purpose or configuration of N-fold cross validation. The first training–validation pair as initially employed to evolve the first FXGP individual, any later calls to evolve a new FXGP individual are relative to the failure criteria triggering retraining. Moreover, retraining results in all current population content being reset as the re-trigger event is taken to imply that the current content is inappropriate.

The trading agent takes the form of a decision tree and corresponding set of *coevolved* technical indicators or DT and TI populations respectively. Evolving the TI provides the opportunity to capture properties pertinent to specific nodes of a decision tree defining the overall trading rule; where the characterization of such temporal properties is known to be significant for a wide range of time series tasks [12]. Four prices – Open, High, Low and Close – are used as inputs to the TI population. Members of the DT population define the trading rule, and it is only with respect to the DT individuals that fitness is evaluated i.e., a symbiotic GP relationship [8,9].

2.2 Training

2.2.1 Initialization

The *TI population* is randomly initialized with (initial) size defined by the user. TI individuals assume a linear GP representation (e.g., [13]) with instruction set summarized by Table 1. Moreover, each TI has a header defining the basic TI properties: type, scale, period, shift (Table 2).

Table 1. TI functions. Note the three forms of division.

Function	Definition	Function	Definition
Addition	$R[x] \leftarrow R[x] + R[y]$	Subtraction	$R[x] \leftarrow R[x] - R[y]$
Multiplication	$R[x] \leftarrow R[x] \times R[y]$	Square root	$R[x] \leftarrow \sqrt{R[y]}$
Division	$R[x] \leftarrow R[x] \div R[y]$	Invert	$R[x] \leftarrow 1 \div R[x]$
Div-by-2	$R[x] \leftarrow R[x] \div 2$	–	–

Table 2. TI parameters. Estimated relative to the current hour t of trading.

Parameter	Description
TI type	Moving Average (MA), Weighted Moving Average (WMA) or Value
TI scale	TI that crosses 0 or TI that crosses price
Period n	Number n of hours in a price history to calculate MA or WMA
Shift m	Price m hours back

TI programs assume a register level transfer language (Table 1). Thus, $R[x]$ denotes the content of register x, $R[y]$ denotes either: register y content; a price, or; a price m hours back in (relative) time. Register $R[0]$ is assumed to contain a TI value after executing a TI program. The MA type of TI is calculated as (1) whereas the WMA type of TI is calculated as (2), where V_j is a TI value.

$$MA_i = \frac{\sum_{j=1}^{n} V_j}{n} \tag{1}$$

$$WMA_i = \frac{\sum_{j=1}^{n} \frac{V_k}{j+1}}{\sum_{j=1}^{n} \frac{1}{j+1}} \tag{2}$$

The *DT population* is initialized to a fixed size as defined the user. A DT header includes the following information: DT *score* in pips, number of trades and a size of a S/L order in pips. The score and number of trades are initialized with 0; whereas the S/L is assumes a fixed interval. A DT consists of a variable number of nodes. Each node consists of a conditional statement with either single or dual antecedent tests of the form:

- *if*$(X_i > Y_i)$ *then else*
- *if*$((X_i > Y_i)$ *and* $(X_{i+m} < Y_{i+m}))$ *then else*

where X_i and Y_i can be 0, price or a TI. The *then* and *else* statements reference either: the next node or one of the trading signals: buy, sell or stay. Thus, a DT population is randomly generated with respect to X_i and Y_i scales; albeit under the following constraints:

1. if X_i is 0, then Y_i can be any TI which crosses 0 and can not be a price or a TI which crosses the price, or;
2. if X_i is price, then Y_i can be also a price or a TI which crosses the price, and can not be 0 or a TI which crosses 0.

Note in the case of a dual antecedent clause, X_{i+m} and Y_{i+m} represent the value of X_i or Y_i respectively, albeit m samples back in (relative) time. FXGP can generate additional TI during DT population initialization if the TI population does not have a TI capable of satisfying the DT initialization constraints.

2.2.2 Fitness and Selection

The DT fitness is defined as a DT score in pips over the training records. When TI and DT populations are initialized, FXGP simulates the trading activity for each DT over the training records and stores the score and the number of trades in the DT header. The number of trades is used to penalize the DT score for too high or low a frequency of trading. Moreover, if a DT generates only *buy* or *sell* signals its score is discarded and the DT targeted for replacement. The subset of DT individuals with lowest scores are explicitly identified for replacement cf., a breeder model of selection / replacement. Thus, a fixed percentage or *gap* size of the DT population is replaced per generation. All variation is asexual, thus following parent selection, cloning and variation takes place where either the DT or TI component of a clone can be varied. Following DT selection, any TI individual that no longer receive a DT index are considered ineffective and are also deleted (resulting in a variable size TI population).

2.2.3 Mutation

FXGP uses mutation to produce offspring. Only one DT node or one linked TI can be mutated in each cloned TI–DT pair. FXGP randomly selects the target for mutation (TI or node) according to the probability of mutation (Table 4). A *mutated TI* is first cloned to avoid interfering with other DT employing the same TI. The following parameters and functions of a TI can be mutated: TI type, period (n), shift (m), generate a new function, delete a function, or insert a function. Likewise, a *DT Mutation* also begins by cloning the parent, and then applies one of the following to define an offspring:

1. Generate a new conditional function;
2. Increment / decrement shift parameter m;
3. Generate new X_i and Y_i;
4. Switch X_i and Y_i;
5. Switch content of *then* and *else* clauses;
6. Insert new *then* clause content.
7. Insert new *else* clause content.

Training stops when the specified number of generations was reached or when the best score in the DT population plateaus for a fixed number of generations, τ (Table 4). Thereafter a validation cycle is initiated.

2.3 Validation

During validation we require a proportion of the population α_v to demonstrate feasible trading behaviour before trading may actually commence. FXGP checks

the score of every tree in a DT population and if the DT score is greater than $\alpha_v \times$ *best score* it tests the DT and increments the tested tree's counter. When all DT are evaluated, FXGP checks the number of the tested DT and if it is greater than $\alpha_v \times$ *DT population size* then the TI–DT pair with best score on validation is selected as the 'champion' for trading, otherwise the entire training procedure is restarted i.e., rather than invest more generations in a population which fails under validation we choose to reinitialize and restart the training cycle.

2.4 Trading and Retraining Criteria

During FX trading, the following three trading quality criteria are monitored and used to trigger retraining from an entirely new population pair of TI–DT populations: 1) Drawdown i.e., decline in account value – see Glossary; 2) The number of consecutive losses permitted; and 3) The number of consecutive hours without trading activity. This approach lets FXGP retrain the DT population only when the market situation is deemed to differ from that of the last training period. In doing so we explicitly recognize the non-stationary nature of the task. Specific values for the quality criteria are established by the user. When the quality criteria are exceeded, FXGP stops trading, reinitializes the TI and DT populations and restarts the training-and-validation cycle. Content of the training and validation partitions is taken relative to the point 't' at which the trading quality criteria interrupted trading. Once a new TI–DT champion is identified trading resumes at the point trading was interrupted.

2.5 Source Data

The historical rates for EURUSD, USDCHF and EURUSD (1 hour resolution) were downloaded with MetaTrader 4 FX terminal from the MetaQuotes Software Corp. history centre [11,14]. Each 24 hour period therefore typically consists of 24 samples. Sampling at the rate of one hour intervals is conducted to reduce the impact of "trading noise" [15]. Three datasets are employed each of which includes 2008-2011 historical rates and consists of following fields: Date, Time, bid price Open (Open), bid price High (High), bid price Low (Low), bid price Close (Close) and Volume.

3 Experimental Setup

Performance is evaluated for the case of TI–DT trading agents as evolved from a single initial cycle of evolution versus the proposed scheme for retraining TI–DT trading agents interactively as established by the trading quality criterion (100 runs in each case); hereafter static and retrain respectively. Each run simulates trading activity for three major currency pairs (EURUSD, USDCHF and EURCHF) over the trading period from January 2, 2009 to December 30, 2011 (approx. 18,500 hours). These pairs are popular during all trading sessions i.e.,

Table 3. Trading conditions with initial trading balance of 100,000 USD

Condition	Value	Condition	Value
Spread USDCHF	0.0003	Min. S/L level	0.0005
Spread EURUSD	0.0002	Pip	0.0001
Spread EURCHF	0.0005	Leverage	1:100

a trading day typically consists of 24 samples. The trading conditions [14] are described in the Table 3.

An initial balance of 100,000 USD is assumed and the leverage is 1:100. The account balance was then recalculated in USD based on a "2% Rule" [16] i.e., only two percent of the current balance may be reinvested at the next round of trading. Moreover, in addition to quoting the resulting pips and USD at the end of 3 years, we also consider the 'overall' trading outcome when conducting trading across a portfolio of currencies. This represents the case of a trader conducting independent runs for all three currency pairs and expressing performance as the combined income from the portfolio of three currency pairs (again at the end of the 3 year period).

Table 4. FXGP parameterization

Parameter	Value	Parameter	Value
DT pop. size	100	DT gap	25
Training period, hours (N_t)	1,000	Validation period, hours (N_v)	500
Max. generation	1,000	Max. DT size, nodes	6
Max. TI program size, steps	8	Number of TI registers	2
Probability of TI mutation	0.5	S/L size, pips	100
Training plateau length (τ)	200	TI–DT validation fraction (α_v)	0.7
Trading Quality Criteria			
Num. consecutive losses	5	Drawdown, pips	400
–	–	Max. time without trading activity, hrs	72

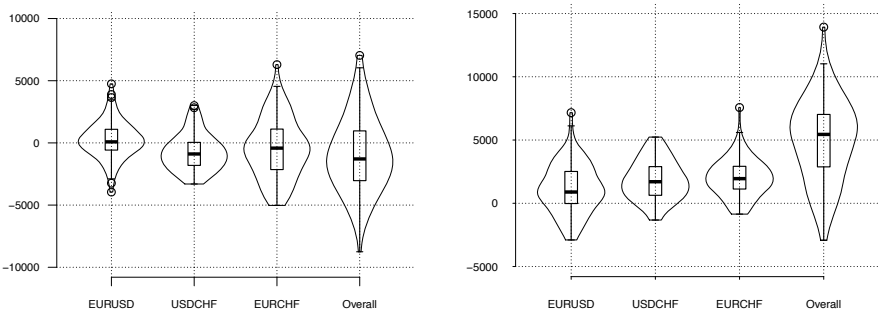
4 Results

The resulting (pips) distribution for one hundred runs for TI–DT under 'static' and 'retrain' deployments are shown in the Fig. 1(a) and Fig. 1(b) respectively. Table 5 provides summary statistics. Not only is the profitability of TI–DT traders with retraining significantly higher, but the number of profitable runs are discovered between 1.3 to 3.5 times more frequently (depending on the currency pair). For completeness the account balance *during trading* of the best, typical and worse runs are shown in Fig 2(a) (pips) and Fig 2(b) (USD).

Finally, we can also consider the typical interval between retraining episodes, Figure 3. Retraining appears to be called at a median rate of 200 hours, implying that over the total (three year) trading interval, there are 92 calls for retraining. From an end user perspective this corresponds to retraining once every 8 days

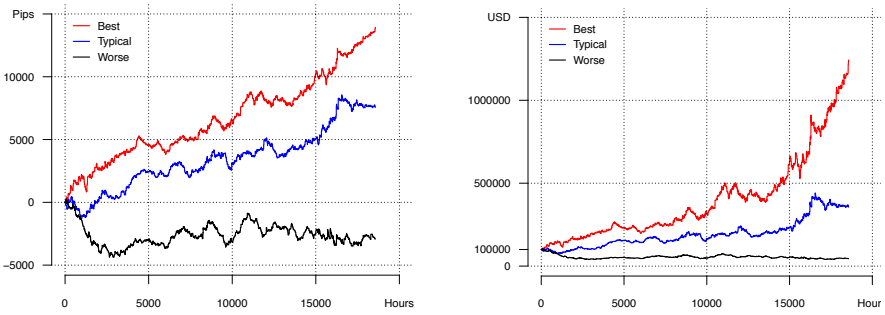
Table 5. TI–DT Static versus Retraining summary. For consistency with Figure 2(b) 'Overall balance' include the initial trading balance of 100,000 USD.

Description	retrain	static	retrain vs static, %
EURUSD profitable runs, %	75	58	129.3
USDCHF profitable runs, %	88	25	352.0
EURCHF profitable runs, %	89	38	234.2
Overall profitable runs, %	92	34	270.6
Best overall score, pips	13929	7036	n/a
Typical overall: pips	7500	-2000	n/a
Worse overall: pips	-2922	-8753	n/a
Best overall balance: USD	1,242,920.54	414,005.96	n/a
Typical overall balance: USD	356,825.19	61,528.97	n/a
Worse overall balance: USD	44,816.85	13,513.47	n/a



(a) Pips (b) USD

Fig. 1. Performance distributions for TI–DT in **pips**: (a) static TI–DP and (b) TI–DP with retraining criteria. Internal box-plot provides quartile statistics. Violin profile characterizes the distribution.



(a) Pips (b) USD

Fig. 2. Adaptive TD results best, typical and worse runs for (a) Pips metric and (b) USD (including initial balance of 100,000 USD)

(with trading performed on a 5 day trading week). Retraining the population under the current parameterization takes 30 seconds on an iMac desktop.[1] This implies that the approach is able to complete the retraining cycle well before the next (1 hour) sample is received, effectively rendering the approach 'real-time' from a deployment perspective.

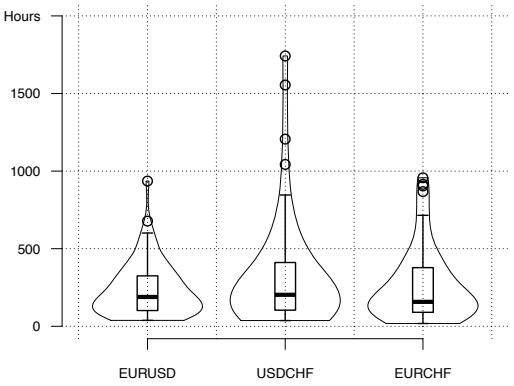


Fig. 3. Distribution of retraining per currency over trading period

Finally, by way of comparison, we note that investing $100,000 in a commercial mutual fund over the same period (January 2, 2009 to December 30, 2011) would result in a best case rate of return of 5.58% [17] i.e., an account balance of $117,687 at the end of the period. This is clearly better than the case of static TI–DP, but is typically bettered by TI–DP with retraining enabled. Moreover, post 2011 hindsight would be necessary to select such a profitable mutual fund.

5 Conclusion

It is increasingly being acknowledged that the non-stationary aspect of trading environments places additional requirements on the model building process for constructing trading agents. There are at least three different factors: providing an appropriate representation, detecting change, and maintaining diversity in the models proposed. This work assumes two specific properties: 1) a highly modular representation care of coevolving TI and DT populations, and 2) dynamically re-triggering training relative to a set of trading criteria or change detection. Moreover, the approach taken to change detection is to adopt criteria that a trader might well assume in practice. Such a scheme appears to be feasible with both profitable trading strategies typically discovered and, given the hourly rate of receiving new data, sufficient time to complete retraining before a new sample is received. As a final verification of the approach, randomly selected

[1] Intel Core i7, 2.8 GHz, 16 Gb RAM 1333 MHz DDR3, OS X 10.7.5.

TI–DT 'traders' were implemented in the MetaTrader 4 trading terminal [14] as Expert Advisors and tested on the FxPro demo account. The obtained result were similar to the results of the FXGP simulation.

Acknowledgements. The authors gratefully acknowledge support from MITACS and NSERC (Canada).

References

1. Morozov, I.V., Fatkhullin, R.R.: Forex: from simple to complex. Teletrade Ltd. (2004)
2. Settlement, B.F.I.: Triennial central bank survey of foreign exchange and otc derivatives market activity - preliminary global results (April 2010), http://www.bis.org/press/p100901.htm
3. Passamonte, A.: Six facts that give forex traders an edge. Forex Journal (2011), http://www.forexjournal.com/fx-education/forex-trading/12125-six-facts-that-give-forex-traders-an-edge.html
4. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. SCI, vol. 194. Springer, Heidelberg (2009)
5. Tuite, C., Agapitos, A., O'Neill, M., Brabazon, A.: A Preliminary Investigation of Overfitting in Evolutionary Driven Model Induction: Implications for Financial Modelling. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Drechsler, R., Farooq, M., Grahl, J., Greenfield, G., Prins, C., Romero, J., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Urquhart, N., Uyar, A.Ş. (eds.) EvoApplications 2011, Part II. LNCS, vol. 6625, pp. 120–130. Springer, Heidelberg (2011)
6. Mayo, M.: Evolutionary Data Selection for Enhancing Models of Intraday Forex Time Series. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 184–193. Springer, Heidelberg (2012)
7. Atwater, A., Heywood, M.I., Zincir-Heywood, A.N.: GP under streaming data constraints: A case for Pareto archiving? In: ACM Genetic and Evolutionary Computation Conference, pp. 703–710 (2012)
8. Lichodzijewski, P., Heywood, M.I.: Symbiosis, complexification and simplicity under GP. In: ACM Genetic and Evolutionary Computation Conference, pp. 853–860 (2010)
9. Doucette, J.A., McIntyre, A.R., Lichodzijewski, P., Heywood, M.I.: Symbiotic coevolutionary genetic programming. Genetic Programming and Evolvable Machines 13(1), 71–101 (2012)
10. Contreras, I., Hidalgo, J.I., Núñez-Letamendia, L.: A GA Combining Technical and Fundamental Analysis for Trading the Stock Market. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 174–183. Springer, Heidelberg (2012)

11. ICM Trade Capital Markets Ltd.: Guide to online forex trading 19 pages
12. Wagner, N., Michalewicz, Z., Khouja, M., McGregor, R.R.: Time series forecasting for dynamic environments: The DyFor genetic program model. IEEE Transactions on Evolutionary Computation 11(4), 433–452 (2007)
13. Brameier, M., Banzhaf, W.: Linear Genetic Programming. Springer (2007)
14. MetaQuotes Software Corp., http://www.fxpro.com/trading/cfd/mt4/forex (accessed September 2012)
15. Investopedia, http://www.investopedia.com/terms/n/noise.asp#axzz27d0d2rid (accessed September 2012)
16. Investopedia, http://www.investopedia.com/terms/t/two-percent-rule.asp#axzz2710QU8jR (accessed September 2012)
17. RBC Global Asset Management: Investment portfolio tools (January 2013), https://services.rbcgam.com/portfolio-tools/public/investment-performance/

Identifying Market Price Levels Using Differential Evolution

Michael Mayo

University of Waikato, Hamilton, New Zealand
mmayo@waikato.ac.nz
http://www.cs.waikato.ac.nz/~mmayo/

Abstract. Evolutionary data mining is used in this paper to investigate the concept of support and resistance levels in financial markets. Specifically, Differential Evolution is used to learn support/resistance levels from price data. The presence of these levels is then tested in out-of-sample data. Our results from a set of experiments covering five years worth of daily data across nine different US markets show that there is statistical evidence for price levels in certain markets, and that Differential Evolution can uncover them.

Keywords: differential evolution, finance, support and resistance.

1 Introduction

Do price levels exist in market series? The idea of price levels – often referred to as "support" and "resistance" – has been prevalent anecdotally for nearly as long as financial markets have existed. These phenomenon are a staple feature in most trading and finance textbooks (e.g. [3]), which typically teach people to buy assets at support lines (because prices are likely to rise from them) and conversely sell assets at resistance lines (because prices are likely to fall). But is there statistical evidence that support and resistance lines exist?

Answering this question is the focus of this paper. We adopt a machine learning-based methodology, utilizing Differential Evolution (DE) [4], in an attempt to learn price levels from market data. We then compare these optimized levels to randomly selected levels in order to determine appreciable differences. Specifically, if a set of levels where price reverses are found in the training/in-sample data, then we want to know if these levels continue to persist in chronologically subsequent testing/out-of-sample data. If the levels do persist, then it can be said that the concepts of support and resistance have foundation. On the other hand, if the best levels found in-sample cannot be used to predict reversals out-of-sample, then we can conclude confidently that the dual concepts of support and resistance have no foundation.

To date, there has been little consideration of this question in the applied machine learning/finance research literature. Most other approaches deal with pattern-based turning point prediction and often ignore absolute prices. This research, on the other hand, focuses on turning points based on absolute price

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 203–212, 2013.
© Springer-Verlag Berlin Heidelberg 2013

levels. The question is whether or not the market "remembers" these old levels and therefore whether history may repeat predictably. Note that the levels used here are basically horizontal lines; we leave the generalization of the method to angled trend lines and channels for future work.

Our results described in this paper show that in some markets, especially indices such as the Dow Jones Industrial Average, the presence of price levels can be detected with statistical significance.

2 Background

In this section, we cover the concept of a "price level" in a little more detail and briefly explain the variant of DE used here.

2.1 Price Levels in Markets

A support or resistance line is, by definition, a point in the market where price has an increased probability of reversal. For the purposes of this paper, a good price level is therefore any price that the market often reaches but infrequently penetrates. To illustrate this concept, consider Figures 1 and 2.

Fig. 1. Example of a poor level with excessive distance from price

Fig. 2. Example of a poor level with excessive penetrations by price

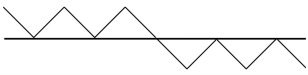


Fig. 3. Example of a strong level with a high ratio of failed to successful penetrations

These figures both depict poor levels. In the first case, price never reaches the level and therefore there is no opportunity for price to reverse at the level. In the second case, price frequently penetrates the level, so the level is clearly not a support or resistance for price.

Figure 3, on the other hand, depicts conceptually a good level. In this case, price frequently reverses at the level, and when it does eventually break through, the level that was formally support now becomes resistance. The same concept is depicted again in Figure 4, this time somewhat more realistically using candlesticks.

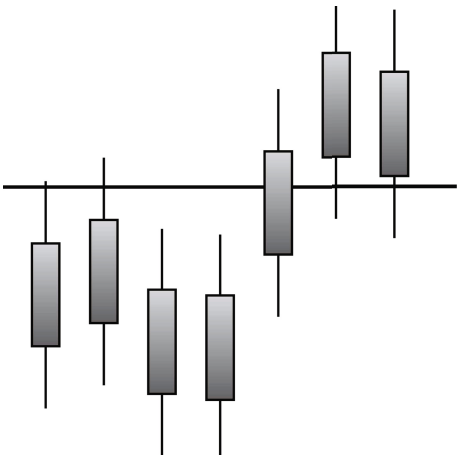


Fig. 4. Figure similar to Fig. 3 but depicted more realistically using candlesticks

In this paper, we define a level as any price point with a high probability of a "failed penetration" on the daily charts. A failed penetration occurs when a candlestick's wick touches the level, but the open and close prices are both on the same side of the level.

Table 1. Key parameters in differential evolution and the values used in this paper for $VS = 10$ as suggested by [2]

Parameter	Value	Definition
VS	10	Individual size (i.e. dimension)
NP	18	Population size
F	0.67	Amplification factor
CR	0.50	Probability of crossover
E	20,000	Max. number of function evals

For example, in Figure 4, the first and last pair of candlesticks depict failed penetrations; however the fifth candlestick is a successful penetration because the open and close prices straddle the level. We can therefore estimate the probability of a failed penetration at the level as $\frac{4}{5}$. This is the essence of how we measure the value of a price level in this paper.

2.2 Differential Evolution

Lack of space prevents a complete description of the DE algorithm here. Essentially, DE is an evolutionary optimization algorithm for continuous spaces first proposed in [4]. The algorithm has been used in financial applications previously [1].

There are two main reasons for using DE in this research. Firstly, DE's proponents claim that the algorithm's global convergence properties are very good, and this has been demonstrated in past studies. Secondly, DE requires very few user-specified parameters. Furthermore, a recent study [2] has elucidated the

best combinations of parameters for different problem sizes. In the experiments described here, the problem size (i.e. the number of dimensions) VS is set to 10, and the corresponding set of parameter values (according to [2]) that we use is given in Table 1.

In all of our experiments, we use the standard *DE/rand/1/bin* variant of DE described in [4].

3 Differential Evolution for Price Level Identification

In this section, the basic adaptation of DE for price level identification is described. We also describe the randomized algorithm used as a control case in the experiments in the next section.

3.1 Individual Representation

Recall that the aim of this research is to use DE to optimize a set of levels, such that the probability of penetration failures at those levels is maximized. The set of levels found would thus correspond to a set of support/resistance levels. Ideally, the size of this set of levels should be variable because, for example, in one market at a particular time there may be many active levels, but in another market (or in the same market but at a different time) there may only be a few active levels.

However, varying the number of levels like this poses difficulties when DE is applied to the problem, because DE expects vectors to be fixed-length. We therefore modify the problem slightly and require that the cardinality of the set of levels be fixed to VS. In other words, a set S of price levels can be defined according to Equation 1. The levels l_0, l_1, etc, can therefore be encoded directly as vector elements for DE to optimize.

$$S = \{l_0, l_1, ..., l_{VS-1}\}, l_i \in \mathbf{R} \tag{1}$$

An issue is that the individual levels, if unconstrained, may be "optimized" to the same single best value. This was in fact the case in early testing of the algorithm: if a single level proved to have high value, then it was usually the case that DE would set l_0, l_1, etc, all to that price, with the net result being that only a single level was discovered.

The solution to this problem is to limit the values of each variable $l_i \in S$ to non-overlapping ranges. Let us assume that the price series being used is divided into a training series T_0 that is followed chronologically by a testing series T_1. T_0 is used for optimization, and T_1 is set aside (it will be used later to calculate the out-of-sample values of the best individual from the T_0 optimization phase).

From the training series T_0, a price range can be straightforwardly calculated, and this range can be divided by the number of levels VS to give a range size per level, as Equation 2 shows.

$$rangeSize = \frac{high(T_0) - low(T_0)}{VS} \tag{2}$$

Once the range is known, then upper and lower bounds on the value of each level can be calculated, Equations 3 and 4 show.

$$lower_i = low(T_0) + (i \times rangeSize) \tag{3}$$

$$upper_i = lower_i + rangeSize \tag{4}$$

We therefore modify the standard DE algorithm and require that:

$$lower_i \leq l_i \leq upper_i, \forall l_i \in S. \tag{5}$$

This is achieved by (i) initializing all new individuals with values for l_i selected uniformly and randomly in the appropriate range and (ii) scoring all offspring individuals with levels outside of the appropriate range with maximum negative value, i.e. such individuals are allowed to be generated by the search process but are immediately "aborted".

3.2 Value Function

Evolutionary search algorithms require a value function that is either maximized or minimized by the search process. In our case, the value function is an estimate of the probability of the failure of price to penetrate the levels in the current set of levels S being considered. Value should therefore be maximized.

Recall that we have a training set T_0 used to estimate value for each set S – and more generally, let T be any series of daily Open-High-Low-Close (OHLC) data.

Given some T and a particular level $l \in S$, let us define two useful functions: $f(l, T)$, specifically the number of failed penetration attempts of price against level l in the series T (i.e. the number of wicks that intersect l where the open and close of the day are both on the same side of the level); and $t(l, T)$, the total number of intersections between price and l in T (i.e. the number of bars in T that overlap l regardless of whether it is only a wick or a full candlestick body).

Obviously, then, it follows that $t(l, T) \geq f(l, T)$ will always hold.

To illustrate these functions, consider Figure 4. If the figure depicts the only touches to the level l in T, then $f(l, T) = 4$ and $t(l, T) = 5$.

Clearly, then, for each level l we will want to maximize the ratio of the first function to the second function in order to find good levels that are likely to "repel" price. Mathematically, this is expressed in the function defined by Equation 6.

$$V(S, T) = \frac{\sum_{l \in S} f(l, T)}{(\sum_{l \in S} t(l, T)) + K} \tag{6}$$

A constant K is added to the denominator in order to reduce the value of levels with only a small number of touches. For example, suppose $VS = 1$, $S = \{l\}$, $f(l, T) = 1$ and $t(l, T) = 1$. Then the ratio of f to g expressed as a percentage is 100%, but the sample size is very small and this result is therefore unreliable.

With K fixed to 10 (which is K's value in all experiments reported here), the value of this level is reduced appropriately to $\frac{1}{11}$. For the level depicted in Figure 4, the value would therefore be $\frac{4}{5+10} = \frac{4}{15}$, which is an underestimate of the true probability of a failed penetration but certainly reasonably greater than $\frac{1}{11}$.

In summary, the optimization problem that we will use DE for can be expressed as the problem of finding a set S_{DE} such that $V(S_{DE}, T_0)$ is maximized, with the out-of-sample value of interest denoted by $V(S_{DE}, T_1)$.

3.3 Control Case

In order to evaluate the efficacy of DE for price level identification, we need something to compare it to. In this research, we consider the case of simply randomly generating the levels subject to the constraints expressed by Equations 1-5. This is performed using the training data T_0 and is used as the control for our evaluation. We refer to such a randomly generated set of levels by S_{RN} and therefore the out-of-sample value (of the control algorithm for a single trial) is $V(S_{RN}, T_1)$.

Note that we also use the exact same method to randomly initialize new individuals at the start of each DE run – thus the key difference being tested is the ability of DE to optimize levels within each level's range.

4 Evaluation

In this section, the details of the evaluation and the results are covered.

4.1 Datasets

Three US stock indices and six US company stock markets were selected for the evaluation. The indices (S & P 500, Dow Jones Industrial Average and the Nasdaq Composite) were chosen because they are representative of the market at large. The six companies were chosen randomly from those that make up the respective indices, the only criteria being that (i) they needed to have at least five year's worth of price data available and (ii) no two companies from the same industry were chosen. Table 2 lists details of the selected markets.

For each market, daily OHLC data for the five years to 3rd August 2012 was downloaded from Yahoo! Finance [5].

4.2 Method

For each of the market data sets listed in Table 2, we conducted 50 trials.

Each trial consisted of (i) selecting a random 1 year subsample from the five year's worth of data, of which the first 6 months was used for training (T_0) and the second six months for testing (T_1);(ii) generating a random set of levels S_{RN} given T_0 which are subject to the constraints specified by Equations 1-4; and

Table 2. Markets analyzed in this study. Market data is daily OHLC bars for five years to 3rd August 2012. Three month volume data from [5] as at 14th August 2012 is also given for the company stock markets, so as to provide an indication of the liquidity of each market.

Symbol	Market	Volume (3m)
SPY	S & P 500	na
DJIA	Dow Jones Industrial Average	na
COMP	Nasdaq Composite Index	na
AAPL	Apple	14,696,400
BA	Boeing Airlines	4,163,520
CELG	Celgene	3,622,660
JEC	Jacobs Engineering Group	955,114
JNJ	Johnson & Johnson	15,793,800
KMB	Kimberly-Clark	2,845,560

(iii), generating an optimized set S_{DE} using differential evolution to maximize the value function on T_0.

The out-of-sample values $Val(S_{RN}, T_1)$ and $Val(S_{DE}, T_1)$ were then computed and recorded.

Thus, a total of nine markets $\times$ 50 trials or 450 experiments were conducted.

4.3 Results

We present the results first graphically using standard box-plots in Figures 5-7 . Each figure depicts the results of applying each algorithm – randomized control and DE – fifty times to a random 1 year sub-sample of each five year market price series. We will then we discuss the statistical significance of the results.

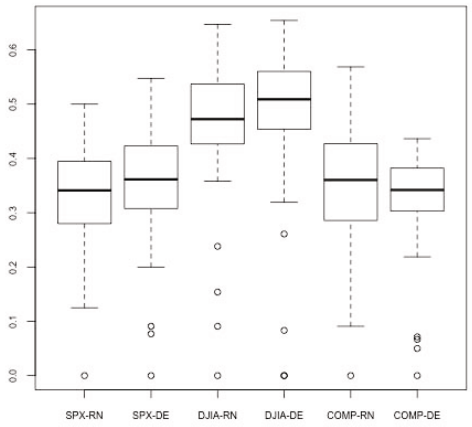


Fig. 5. Market-algorithm (x axis) vs out-of-sample value for best individual (y axis), for indices

Examining firstly the results on the index data in Figure 5, we can see that the DE algorithm does indeed find levels in two out of the three indices that lead to out-of-sample improvements. For example, in the Dow Jones Industrial Average data, median out-of-sample value improves from approximately 0.47 to 0.51. Similarly, the S & P 500 index also appears to exhibit price levels, but the mean out-of-sample value is less: the improvement is from a median of 0.34 to 0.36 instead.

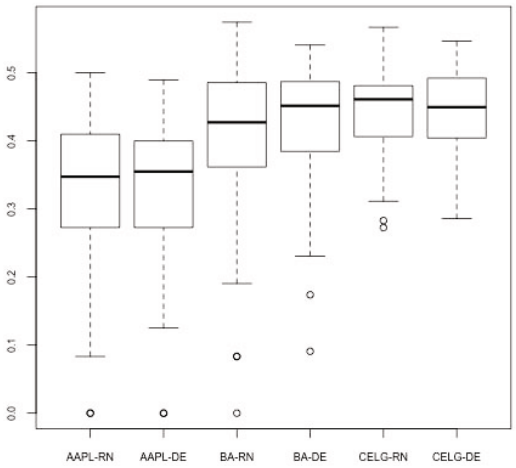


Fig. 6. Market-algorithm (x axis) vs out-of-sample value for best individual (y axis), for stocks 1-3

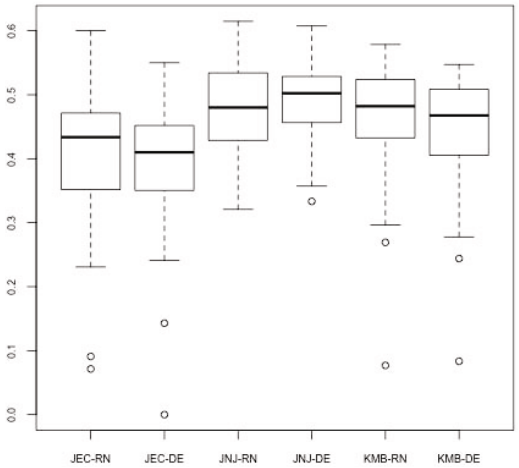


Fig. 7. Market-algorithm (x axis) vs out-of-sample value for best individual (y axis), for stocks 4-6

(Remember that value is a significant underestimate of the true probability of a failed penetration due to the smoothing factor K in the value function.)

The NASDAQ Composite Index, on the other hand, exhibits an out-of-sample experimental decrease in median value from 0.36 to 0.34. However, it should be noted that out that the variance in the level values for this index is quite high in the control case, but when DE is applied, the optimization process decreases variance considerably. In other words, DE makes the level quality more predictable, albeit at the cost of a slight decrease in median value.

Table 3. P-values from paired statistical significance tests comparing the mean (T-Test) and median (Wilcoxon) out-of-sample values of the best individuals obtained using DE compared to randomized level selection. Circles denote significant differences in the means/medians at 95% confidence.

	T-Test	Wilcoxon
SPY	0.062	0.013 •
DJIA	0.003 •	0.003 •
COMP	0.069	0.121
AAPL	0.677	0.478
BA	0.023 •	0.043 •
CELG	0.861	0.905
JEC	0.293	0.255
JNJ	0.071	0.135
KMB	0.104	0.107

Figures 6-7 depict the results of applying the control and DE algorithms to the six stocks. Again, DE results in improvements in some cases (namely, Apple, Johnson & Johnson and Boeing Airlines), but in other cases it fails to improve median value. The probable explanation for this may be market volume: according to Table 2, these are the three stocks with largest volume. The remaining three markets have less volume, therefore less liquidity, and therefore the presence of price levels appears to be more difficult to detect.

Finally, we performed a series of statistical significance tests to verify the results inferred visually from the box plots. For each market, two statistical tests were carried out: a standard T-Test comparing the mean out-of-sample performance of the control algorithm vs DE, and a non-parametric Wilcoxon signed rank test comparing the medians. Both tests are those implemented in the widely used statistical computing package R [6].

The results of these tests are shown in Table 3, where lower p-values indicate increased likelihood that the out-of-sample means/medians over fifty trials are not the same. A number of the tests, particularly those on the Dow Jones Industrial Average data and Boeing Airlines, are significant with 95% confidence (in fact, the DJIA tests are also significant at 99% level). Conversely, the tests show no significant differences for some of the other markets such as Apple and Celgene – although several of the test are *close* to significant.

5 Conclusion

To conclude, this paper has investigated the concept of price levels – anecdotally "support" and "resistance" – in markets. We have used Differential Evolution to learn these levels in markets. The learned levels were then compared to levels selected randomly. The results indicate that in some markets (especially those with higher liquidity) these levels do exist, and their presence can be detected with statistical significant using our approach.

References

1. Brabazon, A., O'Neill, M.: Biologically Inspired Algorithms for Financial Modelling. Natural Computing Series. Springer (2006)
2. Pederson, M.: Good Parameters for Differential Evolution. Hvass Laboratories. Technical Report HL1002 (2010)
3. Pring, M.: Technical Analysis Explained. McGraw-Hill (2002)
4. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359 (1997)
5. http://nz.finance.yahoo.com/
6. R Development Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org

Evolving Hierarchical Temporal Memory-Based Trading Models

Patrick Gabrielsson, Rikard König, and Ulf Johansson

School of Business and Information Technology, University of Borås, Sweden
{patrick.gabrielsson,rikard.konig,ulf.johansson}@hb.se

Abstract. We explore the possibility of using the genetic algorithm to optimize trading models based on the Hierarchical Temporal Memory (HTM) machine learning technology. Technical indicators, derived from intraday tick data for the E-mini S&P 500 futures market (ES), were used as feature vectors to the HTM models. All models were configured as binary classifiers, using a simple buy-and-hold trading strategy, and followed a supervised training scheme. The data set was partitioned into multiple folds to enable a modified cross validation scheme. Artificial Neural Networks (ANNs) were used to benchmark HTM performance. The results show that the genetic algorithm succeeded in finding predictive models with good performance and generalization ability. The HTM models outperformed the neural network models on the chosen data set and both technologies yielded profitable results with above average accuracy.

1 Introduction

One recent machine learning technology that has shown good potential in algorithmic trading is Hierarchical Temporal Memory (HTM). It borrows concepts from neural networks, Bayesian networks and makes use of spatiotemporal clustering algorithms to handle noisy inputs and to create invariant representations of patterns discovered in its input stream. In a previous paper [3], an initial study was carried-out where the predictive performance of the HTM technology was investigated within algorithmic trading of financial markets. The study showed promising results, in which the HTM-based algorithm was profitable across bullish-, bearish and horizontal market trends, yielding comparable results to its neural network benchmark. Although, the previous work lacked any attempt to produce near optimal trading models.

One popular group of optimization methods are evolutionary optimization methods. The simplest evolutionary optimization technique is the genetic algorithm. This paper extends the HTM-based trading algorithm, developed in the previous work, by employing the genetic algorithm as an optimization method. Once again, neural networks are used as the benchmark technology.

2 Background

2.1 Technical Indicators

This study uses technical indicators, derived from the E-mini S&P 500 futures market (ES), as features vectors to the HTM models. Technical indicators are commonly used

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 213–222, 2013.
© Springer-Verlag Berlin Heidelberg 2013

in technical analysis, which is the forecasting of future price movements based on an examination of past price movements. To aid in the process, technical charts and technical indicators are used to discover price trends and to time market entry and exit. Technical analysis has its roots in Dow Theory, developed by Charles Dow in the late 19th century and later refined and published by William Hamilton in the first edition (1922) of his book *"The Stock Market Barometer"*. Robert Rhea developed the theory even further in *"The Dow Theory"*, first published in 1932. Modern day technical analysis [12] is based on the tenets from Dow Theory; prices discount everything, price movements are not totally random and the only thing that matters is *what* the current price levels are. The reason *why* the prices are at their current levels is not important.

2.2 Predictive Modeling

A common way of modeling financial time series is by using predictive modeling techniques [17]. The time series, consisting of tick data, is usually aggregated into bars of intraday-, daily-, weekly-, monthly- or yearly data. From the aggregated data, features (attributes) are created that better describe the data, for example technical indicators. The purpose of predictive modeling is to produce models capable of predicting the value of one attribute, the dependent variable, based on the values of the other attributes, the independent variables. If the dependent variable is discrete, the modeling task is categorized as classification. Classification is the task of classifying objects into one of several predefined classes by finding a classification model capable of predicting the value of the dependent variable using the independent variables.

One common approach to solving a classification problem splits the data set into a training-, validation- and test set. The training set is used to train a classification model, while the validation set is used to determine the performance of the classifier and when to stop training in order to avoid over-fitting the model. The model is subsequently applied to the test set to determine the performance of the classifier on previously unseen data, i.e. its generalization ability.

This paper uses a modified *k*-fold cross validation technique, in which the data set is partitioned into *k* folds. Classification models are then trained on *k-2* folds, where the two remaining folds are used as the validation set and the test set respectively. This procedure is then repeated, using a different fold as the validation set and test set in each run. The performance of the classification models are then averaged over all runs to produce a final performance measure.

2.3 Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a machine learning technology based on memory-prediction theory of brain function described by Jeff Hawkins in his book On Intelligence [8] and are modeled after the structure and behavior of the mammalian neocortex. Just as the neocortex is organized into layers of neurons, HTMs are organized into tree-shaped hierarchies of nodes, where each node implements a common learning algorithm [5].

The input to any node is a temporal sequence of patterns. Bottom layer nodes sample simple quantities from their input and learn how to assign meaning to them in the form of *beliefs*. A belief is the probability that a certain cause of a pattern in the input stream is currently being sensed by the node. As the information ascends the tree-shaped hierarchy of nodes, it incorporates beliefs covering a larger spatial area over a longer temporal period. Higher-level nodes learn more sophisticated causes of patterns in the input stream [9].

Frequently occurring sequences of patterns are grouped together to form temporal groups, where patterns in the same group are likely to follow each other in time. This contraption, together with probabilistic and hierarchical processing of information, gives HTMs the ability to predict what future patterns are most likely to follow currently sensed input patterns. This is accomplished through a top-down procedure, in which higher-level nodes propagate their beliefs to lower-level nodes in order to update their belief states, i.e. conditional probabilities [6-7].

HTMs use Belief Propagation (BP) to disambiguate contradicting information and create mutually consistent beliefs across all nodes in the hierarchy [14]. This makes HTMs resilient to noise and missing data, and provides for good generalization behavior.

With support of all the favorable properties mentioned above, the HTM technology constitutes an ideal candidate for predictive modeling of financial time series, where future price levels can be estimated from current input with the aid of learned sequences of historical patterns.

In 2005 Jeff Hawkins co-founded the company Numenta Inc, where the HTM technology is being developed. Numenta provide a free legacy version of their development platform, NuPIC, for research purposes. NuPIC version 1.7.1 [13] was used to create all models in this study.

3 Related Work

Following the publication of Hawkins' memory-prediction theory of brain function [8], supported by Mountcastle's unit model of computational processes across the neocortex [11], George and Hawkins implemented an initial mathematical model of the Hierarchical Temporal Memory (HTM) concept and applied it to a simple pattern recognition problem as a successful proof of concept [4]. This partial HTM implementation was based on a hierarchical Bayesian network, modeling invariant pattern recognition behavior in the visual cortex.

An independent implementation of George's and Hawkins' hierarchical Bayesian network was created in [18], in which its performance was compared to a backpropagation neural network applied to a character recognition problem. The results showed that a simple implementation of Hawkins' model yielded higher pattern recognition rates than a standard neural network implementation.

In [19], the HTM technology was benchmarked with a support vector machine (SVM). The most interesting part of this study was the fact that the HTM's performance was similar to that of the SVM, even though the rigorous preprocessing of the raw data was omitted for the HTM. In [2] a novel study was conducted, in which the HTM technology was employed to a spoken digit recognition problem with promising results.

The HTM model was modified in [10] to create a Hierarchical Sequential Memory for Music (HSMM). This study is interesting since it investigates a novel application for HTM-based technologies, in which the order and duration of temporal sequences of patterns are a fundamental part of the domain. A similar approach was adopted in [15] where the topmost node in a HTM network was modified to store ordered sequences of temporal patterns for sign language recognition.

In 2011 Fredrik Åslin conducted an initial evaluation of the HTM technology applied to algorithmic trading [1]. In [3], the predictive performance of the HTM technology was investigated within algorithmic trading of financial markets. The study showed promising results, in which the HTM-based algorithm was profitable across bullish-, bearish and horizontal market trends, yielding comparable results to its neural network benchmark.

4 Method

4.1 Data Acquisition and Feature Extraction

The S&P (Standard and Poor's) 500 E-mini index futures contract (ES), traded on the Chicago Mercantile Exchange's Globex electronic platform, was chosen for the research work. Intra-minute data was downloaded from Slickcharts [16] which provides free historical tick data for E-mini contracts. Two months worth of ES market data (5^{th} July – 2^{nd} September 2011) was used to train, validate and test the HTM classifiers using a cross-validation approach.

The data was aggregated into one-minute epochs (bars), each including the open-, high-, low- and close prices, together with the 1-minute trade volume. Missing data points, i.e. missing tick data for one or more minutes, was handled by using the same price levels as the previously existing aggregated data point.

A set of 12 technical indicators were extracted from the aggregated data in order to populate the feature vectors for the classification models (the default parameter values used, in minutes, are enclosed within parentheses); Percentage Price Oscillator (12,26), Percentage Price Oscillator Signal Line (9), PPO Histogram (12,26,9), Relative Strength Indicator (14), William's %R (14), Normalized Volatility Indicator (10 and 20), Chaikin Money Flow (20), Bollinger Band %B (20,2), Rate Of Change (12), Fast Stochastic Oscillator (14), Fast Stochastic Oscillator Signal Line (3).

4.2 Dataset Partitioning and Cross Validation

The dataset was split into a number of folds in order to support a cross-validation approach. This was accomplished by creating a large enough window of size $N=22533$ to be used as the initial training set, followed by partitioning the remaining dataset into $k=5$ folds of size $M=7035$. The model was then trained using the training window and validated on *the closest fold ahead of the training window*. This was done for all individuals through a number of generations using the genetic algorithm. The fittest individual from the final generation was then tested on *the closest fold ahead of the validation fold*. Next, the window of size N was rolled forward M data points to include the first fold in the training set and, hence, omitting the oldest M

data points of the initial window. The model was then trained on the second window of size *N*, validated on the next fold ahead of the training window and tested on the closest fold ahead of the validation fold. This procedure was repeated until *k-1* folds had been used as the validation set once and *k-1* folds had been used as the test set once. The performance measure obtained from all *k-1* validations was then averaged to yield an overall performance measure for the model, similar to normal *k*-fold cross validation. Similarly, the performance measure obtained from all *k-1* test folds was averaged to yield an overall performance measure for the fittest models on the test set. Finally, since the genetic algorithm is non-deterministic, the whole procedure was repeated three times and a final averaged performance measure was obtained by averaging the averaged performance measures from all three individual iterations.

The training, validation and test datasets are shown in Fig.1. The initial training window (training window 01) is show to the far left in the figure, followed by validation window 01 and test window 01. Validation and test windows 02-04 are also depicted in the figure and the range of training windows 01-04 are shown along the bottom of the figure. The lighter patch to the left in the figure shows the buffer used to get the moving average calculations going for the technical indicators.

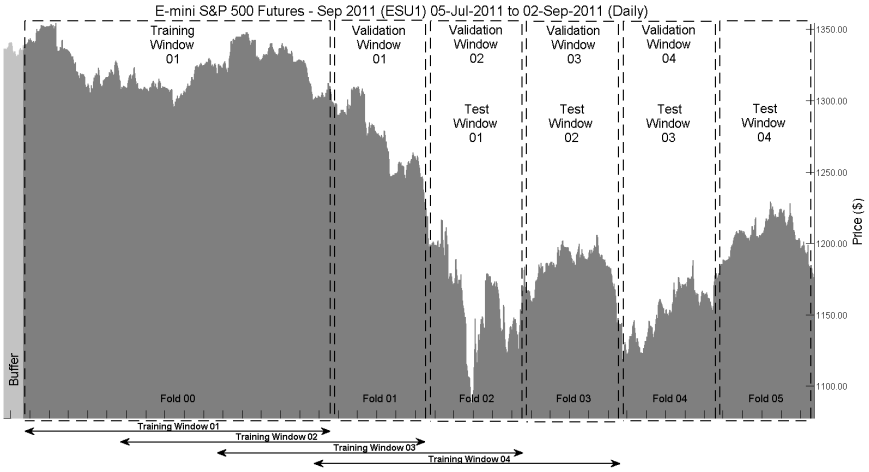


Fig. 1. The dataset for the E-mini S&P 500

4.3 Classification Task and Trading Strategy

The classification task was defined as a 2-class problem:

- Class 0 - The price will not rise >= 2 ticks at the end of the next ten-minute period.
- Class 1 - The price will rise >= 2 ticks at the end of the next ten-minute period.

A tick is the minimum amount a price can be incremented or decremented. For the E-mini S&P 500, the minimum tick size is 0.25 points, where each tick is worth $12.50 per contract. A simple buy-and-hold trading strategy was adopted; if the market is predicted to rise with at least 2 ticks at the end of the next 10-minute period, then buy 1 contract of ES, hold on to it, and then sell it at the end of the period.

4.4 Performance Measure and Fitness Function

The PNL (profit and loss) was chosen as the performance measure. Furthermore, trading fees consisting of $3 per contract and round trip were deducted to yield the final PNL, i.e. $3 for every true positive or false positive. This performance measure was also used as the fitness function for the genetic algorithm. In order to accommodate negative PNLs, tournament selection was used.

4.5 Experiments

The set of feature vectors were fed to the HTM's sensor, whereas the set of class vectors were fed to the category sensor during training as shown in Fig 2. The top-level node receives input from the hierarchically processed feature vector and from the class vector via the category sensor. The output from the top node is class 0 or 1.

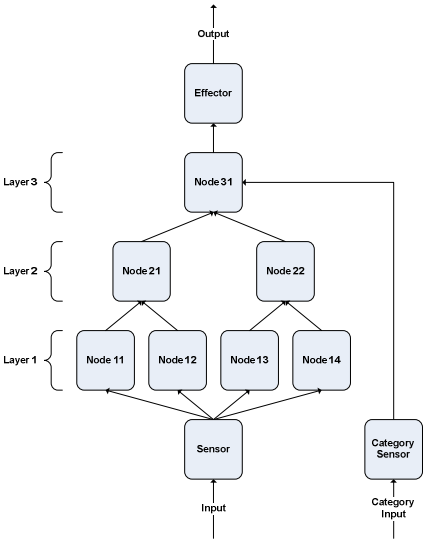


Fig. 2. HTM network configured as a classifier

An HTM network has a number of different parameters that can be optimized. The most important parameters, shown in Table 1, were chosen for the optimization task.

The resulting HTM chromosome consisted of a bit string of length 274. The genetic algorithm was configured to use tournament selection with a tournament size of 4, single-point crossover with a 80% crossover probability and bit-flip mutation with a 4% mutation probability. A population of 50 individuals were evolved over 20 generations and elitism was used.

Each time the genetic algorithm requested a fitness score from an individual, the following sequence of events took place; firstly, an HTM network was created with the parameter settings derived from the individual's bit string pattern. Next, the HTM was trained on the data points contained within the training window and validated on the corresponding validation fold. Each training and validation step produced a performance measure consisting of the PNL subtracted by the total trading cost

Table 1. HTM Parameters

Parameter	Description
Network Topology	Number of layers, including number of nodes per layer
Coincidences	The maximum number of spatial patterns that can be learned
Max Distance	Max. Euclidean distance for clustering spatial patterns to centroids
Sigma	Sigma for the radial-basis function in Gaussian inference mode
SP Algorithm	Spatial Pooler's inference algorithm {Gaussian, K^{th} Root Product}
Groups	The maximum number of temporal patterns that can be learned
Complexity Model	Complexity of the sequence model built by the sequencer [0 - 1.0]
Window Count	Number of windows over which the sequencer will build a model
Window Length	Samples per window where the sequencer will build a model
TP Algorithm	Temporal Pooler's inference algorithm {Max Prop, Sum Prop, TBI}
Transition Memory	How far back in time to look for temporal transitions

(calculated as $3 per trade including brokerage-, transaction- and exchange fees). At the end of each generation, each individuals' performance measure was used as the individual's fitness value by the genetic algorithm.

The neural network benchmark consisted of a recurrent network with feedback loops from the hidden layers. The number of hidden layers, nodes per hidden layer, the learning rate parameter and the momentum were encoded in a chromosome and optimized by the genetic algorithm. Network permutations were left entirely to the discretion of the genetic algorithm, but restricted to a maximum of two layers. The backpropagation learning algorithm was used to update the network weights during training. Sigmoid activation functions were used for all neurons except the output layer which used a softmax activation function. The resulting chromosome had a bit length of 25. The other settings for the genetic algorithm were the same as for the HTM models.

Both technologies were run though Python code, where the execution time requirements for the HTM models were more demanding than the neural networks.

5 Results

The optimization results show that a three-layer HTM was preferred over a two- or four-layer topology. It was also obvious that the Gaussian inference algorithm was preferred for the spatial pooler in every node. Furthermore, a Sigma value close to 1 standard deviation worked best together with the Gaussian inference algorithm. The temporal pooler algorithm used Time Based Inference (TBI) for half the models on average, whereas the maxProp algorithm was preferred for Flash Inference. The remaining parameters settings varied quite a lot amongst the various models.

All models showed positive PNLs in all datasets with minor differences between the validation- and test sets on average. The differences between the validation- and test sets for model accuracy and precision were also insignificant. There was a small loss in accuracy between the training set and the validation set and a noticeable decay in precision. One interesting observation is that even though the number of bad trades (False Positives) outnumber the number of good trades (True Positives) in both the

validation- and test sets, all models still yielded positive PNLs. This suggests that the models identified major increases in price levels correctly. The averaged performance measures for each iteration are shown in Table 2, together with the total averaged performance measures (arithmetic mean for all three iterations).

Table 2. HTM Average Performance

Dataset	TP	TN	FN	FP	Accuracy	Precision	PNL
ITERATION 1							
Training	1464	14897	6066	101	0.726	0.921	$111933.00
Validation	126	3633	2212	173	0.612	0.425	$1958.50
Test	128	3602	2201	213	0.607	0.491	$873.00
ITERATION 2							
Training	1307	14874	6223	124	0.719	0.890	$97762.50
Validation	129	3633	2209	173	0.612	0.412	$2165.00
Test	113	3633	2216	182	0.610	0.439	$1023.50
ITERATION 3							
Training	985	14746	6545	252	0.698	0.832	$58020.00
Validation	101	3640	2237	166	0.609	0.397	$1386.00
Test	93	3656	2236	159	0.610	0.364	$473.00
TOTAL AVERAGE (ALL ITERATIONS)							
Training	1252	14839	6278	159	0.714	0.881	$89238.50
Validation	119	3635	2219	171	0.611	0.411	$1836.50
Test	111	3630	2217	185	0.609	0.431	$790.00

Table 3. ANN Average Performance

Dataset	TP	TN	FN	FP	Accuracy	Precision	PNL
ITERATION 1							
Training	200	14714	7333	288	0.662	0.399	$555.00
Validation	50	4274	2644	64	0.615	0.636	$1507.00
Test	35	4295	2645	57	0.616	0.555	$437.50
ITERATION 2							
Training	45	14936	7488	66	0.665	0.410	$499.00
Validation	14	4324	2681	13	0.617	0.646	$872.00
Test	12	4335	2669	17	0.618	0.503	$135.00
ITERATION 3							
Training	91	14894	7442	107	0.665	0.432	$4174.00
Validation	19	4316	2675	22	0.617	0.499	$773.00
Test	7	4342	2673	11	0.618	0.434	$140.00
TOTAL AVERAGE (ALL ITERATIONS)							
Training	112	14848	7421	154	0.664	0.413	$1742.50
Validation	28	4305	2667	33	0.616	0.594	$1051.00
Test	18	4324	2662	29	0.617	0.497	$237.50

For the ANNs, the genetic algorithm (GA) favored a single layer of hidden neurons. It also produced a higher number of neurons in the hidden layers as compared to the number of inputs. A low learning rate and momentum was selected by the GA.

The PNL was positive in all datasets except two test sets and the difference in model accuracy and precision between the various datasets was insignificant. Despite many ANN models having more bad trades (False Positives) than good trades (True Positives), the PNL was positive for all models but 2 when applied to the test sets. As with the HTM models, the ANN models identified major increases in price levels correctly. The average performance measures are shown in Table 3. All average PNLs are positive and model accuracy and precision are similar to each other in all datasets.

The results show that the HTM models outperformed the neural network models, yielding 2-8 times as much in PNL. Both technologies had above average accuracy in all datasets, whereas the precision was slightly below average in the validation- and test sets. Overall, both models were profitable. The positive PNLs in both the validation- and test sets suggest that both technologies produced models with good generalization abilities.

6 Conclusion

The results show that the genetic algorithm succeeded in finding predictive models with good performance and generalization ability. Although the HTM models outperformed the neural networks with regards to PNL, both technologies yielded profitable results with above average accuracy. The optimization results show that the HTM models prefer a 3-layer network topology with a variable max distance setting and number of coincidences and groups in each layer when applied to the E-mini S&P 500. A Gaussian inference algorithm with a Sigma setting close to 1 standard deviation was preferred by the spatial pooler in all nodes in the network. With respect to the temporal pooler's inference algorithm, flash inference and time-based inference were equally prevalent amongst the nodes. The neural network preferred a single hidden layer with a node count close to the dimensionality of its input and small values for its learning rate and momentum.

References

1. Åslin, F.: Evaluation of Hierarchical Temporal Memory in algorithmic trading, Department of Computer and Information Science, University of Linköping (2010)
2. Doremale, J.V., Boves, L.: Spoken Digit Recognition using a Hierarchical Temporal Memory, Brisbane, Australia, pp. 2566–2569 (2008)
3. Gabrielsson, P., Konig, R., Johansson, U.: Hierarchical Temporal Memory-based algorithmic trading of financial markets. In: 2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), pp. 1–8 (2012)
4. George, D., Hawkins, J.: A hierarchical Bayesian model of invariant pattern recognition in the visual cortex, in. In: Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN 2005, vol. 3, pp. 1812–1817 (2005)

5. George, D., Jaros, B.: The HTM Learning Algorithms. Numenta Inc. (2007), http://www.numenta.com/htm-overview/education/Numenta_HTM_Learning_Algos.pdf
6. George, D., et al.: Sequence memory for prediction, inference and behaviour. Philosophical Transactions - Royal Society. Biological Sciences 364, 1203–1209 (2009)
7. George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLoS Computational Biology 5, 1000532 (2009)
8. Hawkins, J., Blakeslee, S.: On Intelligence. Times Books (2004)
9. Hawkins, J., George, D.: Hierarchical Temporal Memory - Concepts, Theory and Terminology. Numenta Inc. (2006), http://www.numenta.com/htm-overview/education/Numenta_HTM_Concepts.pdf
10. Maxwell, J., et al.: Hierarchical Sequential Memory for Music: A Cognitive Model. International Society for Music Information Retrieval, 429–434 (2009)
11. Mountcastle, V.: An Organizing Principle for Cerebral Function: The Unit Model and the Distributed System, pp. 7–50. MIT Press (1978)
12. Murphy, J.: Technical Analysis of the Financial Markets, NY Institute of Finance (1999)
13. Numenta. NuPIC 1.7.1, http://www.numenta.com/legacysoftware.php
14. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)
15. Rozado, D., Rodriguez, F.B., Varona, P.: Optimizing Hierarchical Temporal Memory for Multivariable Time Series. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part II. LNCS, vol. 6353, pp. 506–518. Springer, Heidelberg (2010)
16. Slickcharts, E-mini Futures (2012), http://www.slickcharts.com
17. Tan, P.N., et al.: Introduction to Data Mining. Addison Wesley (2009)
18. Thornton, J., Gustafsson, T., Blumenstein, M., Hine, T.: Robust Character Recognition Using a Hierarchical Bayesian Network. In: Sattar, A., Kang, B.-H. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 1259–1264. Springer, Heidelberg (2006)
19. Thornton, J., Faichney, J., Blumenstein, M., Hine, T.: Character Recognition Using Hierarchical Vector Quantization and Temporal Pooling. In: Wobcke, W., Zhang, M. (eds.) AI 2008. LNCS (LNAI), vol. 5360, pp. 562–572. Springer, Heidelberg (2008)

Robust Estimation of Vector Autoregression (VAR) Models Using Genetic Algorithms

Ronald Hochreiter and Gerald Krottendorfer

Department of Finance, Accounting and Statistics,
WU Vienna University of Economics and Business
ronald.hochreiter@wu.ac.at, gerald@krottendorfer.eu

Abstract. In this paper we present an implementation of a Vector autoregression (VAR) estimation model using Genetic Algorithms. The algorithm was implemented in R and compared to standard estimation models using least squares. A numerical example is presented to outline advantages of the GA approach.

Keywords: Genetic algorithms, Time series, Vector autoregression (VAR).

1 Introduction

Vector autoregression (VAR) models have been advocated by Sims [9] for application in various fields of Finance, as those models provide a theory-free method to estimate economic relationships. In this paper, we present an implementation of a VAR estimation and simulation for predicting future asset returns using a specially designed Genetic Algorithm and compare the properties of the solution to a model estimated using standard methods. For comparison we apply the model to the Infineon stock. This paper is organized as follows: Section 2 provides an introduction to Vector autoregression (VAR) models. Section 3 outlines theoretical details of the implementation of VAR models using Genetic Algorithms, while section 4 discusses its practical implementation. Section 5 provides numerical results, and Section 6 concludes the paper.

2 Vector Autoregression (VAR) Models

We use a VAR model to predict the next day return of a reference asset, with the VAR model to be described as follows:

$$\beta_1 \cdot z_{a1,l1} + \beta_2 \cdot z_{a2,l2} + ... + \beta_m \cdot z_{am,lm} + ... + \beta_M \cdot z_{aM,lM} = \widehat{y}_T \qquad (1)$$

In this formula $\widehat{y}_T$ is the predicted next day return at $t = T$ and $z_{am,lm}$ are the selected lags from asset am with lag index lm and β_m are appropriate weighting coefficients. Note that samples $z_{am,lm}$ in Equation (1) do not necessarily appear in any consecutive order but are an unordered list of all asset samples that fulfill

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 223–233, 2013.
© Springer-Verlag Berlin Heidelberg 2013

a certain selection criterion. Establishing a VAR model can be broken up in the following steps. Firstly one has to provide a portfolio of assets, whose most appropriate lags are selected for being part of the VAR model (lag selection). In a next step weighting coefficients, one for every selected lag, need to be estimated (weighting coefficient estimation) to create the VAR estimation model to forecast the next days return of the reference asset according to equation (1).

2.1 Asset Selection

We have chosen the IFX stock (Infineon Technologies AG) to be the reference asset y_t. In general, every asset that exhibits an economic relationship to Infineon qualifies to be a relevant assets for the *VAR model portfolio*. These are stocks of IFX stakeholders, such as customers, strategic partners, suppliers or competitors of Infineon. Note that the IFX stock itself is also included in the VAR asset portfolio. These assets have been manually derived from Infineon's financial reports (quarterly and annual reports 2010/2011)[1] and from a market analysis. Furthermore the portfolio includes assets that reflect the actual economic situation of Infineon such as exchange rates or relevant stock indices. A separate script checks all assets and rejects those which are inconsistent. An asset is flagged to be inconsistent, when its daily return exceeds a certain value, when samples are lost or when it is not traded for a longer period.

2.2 Lag Selection

We have adopted the cross correlation function (ccf) as a measure of similarity of an asset of the portfolio against the reference asset as a function of a time-lag applied to it. The ccf value $\rho_{r,am}(lm)$ of asset number am against the reference asset y_t at lag lm is given by

$$\rho_{y,am}(lm) = \frac{E\left[(y_{t-lm} - \mu_y)(z_{am,t} - \mu_{am})\right]}{\rho_y \cdot \rho_{am}} \tag{2}$$

where μ_y and μ_{am} are the means of the reference asset and the comparison asset respectively, and ρ_y and ρ_{am} are their standard deviation values. Furthermore $z_{am,t}$ is the return time series of asset am and y_t of the reference asset accordingly. For practical implementations μ_y, μ_{am}, ρ_y and ρ_{am} have to be estimated by observing samples of the involved time series within a limited time period T_{ccf}. Similarly, the expectation E in Eq. (2) will be applied within the time period T_{ccf}.

Since a large ccf value at lag lm expresses a strong correlation of the reference asset to a lag-shifted asset, we assume that this indicates a strong correlation of the actual sample of the reference asset and the lag-shifted (by lag lm) time sample of this asset. Hence we assume those lags which have a sufficiently large corresponding ccf values $\rho_{r,am}(lm)$ to be *strong lags* and select those to be used for the VAR estimation. We have tested two selection mechanisms:

[1] `http://www.infineon.com/cms/de/corporate/nvestor/ireporting/reporting.html`

- Constant lag number: select M strongest lags out of all past samples of all assets of the portfolio, with M being a preselected value.
- Constant threshold: select all lags, whose corresponding ccf value exceeds a preselected threshold level th. A suitable threshold value has been defined by Tiao and Box [10]:

$$th = \pm \frac{2}{\sqrt{T_{ccf}}} \tag{3}$$

2.3 Weighting Coefficient Calculation

The coefficients will be computed with two methods: a least squares calculation (LS model) and by using a genetic algorithm (GA model). Both methods fit the weighting coefficients to generate N past sample estimations of $\widehat{y}_t$ using Eq. (1) as described in (4) to fit to their corresponding known historical values of y_t.

$$\begin{gathered}
\beta_1 \cdot z_{a1,l1-1} + ... + \beta_m \cdot z_{am,lm-1} + ... + \beta_M \cdot z_{aM,lM-1} = \widehat{y}_{T-1} \approx y_{T-1} \\
\beta_1 \cdot z_{a1,l1-2} + ... + \beta_m \cdot z_{am,lm-2} + ... + \beta_M \cdot z_{aM,lM-2} = \widehat{y}_{T-2} \approx y_{T-2} \\
\cdots \\
\beta_1 \cdot z_{a1,l1-N} + ... + \beta_m \cdot z_{am,lm-N} + ... + \beta_M \cdot z_{aM,lM-N} = \widehat{y}_{T-N} \approx y_{T-N}
\end{gathered} \tag{4}$$

The system of linear equations (4) may be written in matrix notation.

$$B \cdot Z = \widehat{Y} \tag{5}$$

where B is a vector containing the weighting coefficients β_m, Z is a matrix containing lags $z_{am,lm}$ and $\widehat{Y}$ is a vector containing the reference asset sample estimates y_{T-n}.

We will calculate B by adopting a GA optimization that uses a fitness function which penalizes large differences of past estimates of $\widehat{y}_t$ and their corresponding actual past samples y_t:

$$fitness = \frac{1}{N} \sum_{t=T-1}^{T-N} (y_t - \widehat{y}_t)^2 = optimization\ error \tag{6}$$

Finally, the winning chromosome of the GA optimization is the coefficient vector B, since it produces the smallest fitness function result, which we will refer to as the *optimization error*. Note that solving equation (5) does not consider any statistical variance in the VAR process an hence constitute an unrealistic model of a real world process. However, we will address this issue in section (3) and will deliver the mathematical motivation to use the GA optimization for Eq. (5).

For the LS method we have to consider the VAR process model [8]:

$$Y = B \cdot Z + U \tag{7}$$

According to (7), the actual past sample vector Y may be expressed by multiplying the coefficient vector B with lag matrix Z and adding an additional noise vector U, that reflects the statistical behavior of the VAR process. The noise

vector U equals the residual $Y - \widehat{Y}$, hence the LS method can be applied by minimizing the squared noise vector $U \cdot U'$. [2] The solution to this optimization problem eventually delivers the LS optimal coefficient vector $\widehat{B}$:

$$\widehat{B} = YZ'(ZZ')^{-1} \tag{8}$$

Finally, we may use the coefficients, calculated by either the LS or the GA, to forecast the next days yield using (1).

2.4 Robustness

Consider an economic shock that has a relevant and unforeseeable impact on the time series of the VAR model portfolio (*exogenous shock*). Forecasts based on past samples will deliver poor forecasts, since samples prior to the shock do not reflect the actual time series trend anymore. However, the estimation performance will be restored, after the estimation model will solely access samples that occurred past the shock. Consequently, a VAR model which needs fewer past samples for coefficient estimation will recover faster, being more robust to such shocks.

The dimension of matrix Z is $[M \times N]$, with M being the number of selected lags and N being the number of equations in (4). N also determines, how many samples of the past are observed and hence determines how fast a model recovers from an exogenous shock.

- GA model: requires a squared matrix, and therefore N equals M, see Section 3 below.
- LS model: To deliver useful results, the dimension of lag matrix Z needs to be non-square, with $N > M$ [3]. Here, the number of observed past samples may be expressed by $N = M + k$, with k being an additional factor that expresses how much the equation system (7) is over-determined.

In addition, one needs to add the maximal delay of lags (largest lm) resulting in a total number of past lags L required for the estimation model:

$$\begin{aligned} &\text{GA model: } L_{GA} = M + max(lm) \\ &\text{LS model: } \ L_{LS} = M + k + max(lm) \end{aligned} \tag{9}$$

Since the GA approach requires fewer past samples for coefficient estimation, it will require less time to recover after an exogenous shock, and hence will be more robust.

[2] The tickle symbol at U' and Z' in equation (8) designates the matrix transpose operand.

[3] If $N < M$, Eq. (8) does not have a solution at all. In case of a squared matrix ($N = M$), Eq. (8) simplifies to $\widehat{B} = YZ^{-1}$ which reflects the trivial solution of Eq. (7) with $U = O$ (O is the null-vector). A solution that neglects the existence of the noise value U does not deliver realistic values for the sought coefficients, hence the estimation quality will suffer. In fact, the larger N is, the better is the quality of an LS estimation.

3 A New Approach Using Genetic Algorithms

Genetic algorithms and other biologically inspired algorithms have been applied to various time series estimation problems successfully, see especially [3], but also e.g. [11], [2], [7], [5], [1], [6] and [4] among others.

To explain the GA-based VAR method, let us assume the hypothetical case, that the noise vector U of the VAR process is zero. We furthermore assume the lag matrix Z to be square. The VAR model is then defined by:

$$Y = B^0 \cdot Z \tag{10}$$

Although we could easily solve (10) analytically, we want to apply an optimizing algorithm to solve this equation. Furthermore, although we assume that the optimization error would converge to zero, we prematurely stop the iterative optimization process. This causes the estimations $\widehat{Y}$ to substantially differ from Y, which in equation (11) is expressed by the residual vector W:

$$\widehat{Y} = \widehat{B}^0 \cdot Z + W \tag{11}$$

Since we stopped the iterative optimization process before it has converged sufficiently, $\widehat{B}^0$ and B^0 will differ substantially. Let us now assume, that we are able to control the optimization algorithm such that the residual vector W is equal to the noise vector U of the real world VAR model described in (7). In this case $\widehat{B}^0$ of the hypothetical model with noise vector set to zero is identical with the coefficient vector B of the real world VAR model.

$$\begin{array}{ll} \text{real world model:} & B \cdot Z + U \\ \text{hypothetical model:} & B^0 \cdot Z \end{array} \tag{12}$$

$$W = U \Rightarrow \quad \widehat{B}^0 \cdot Z + W = \widehat{B}^0 \cdot Z + U \Rightarrow \widehat{B}^0 = B$$

Now we obtain the estimated sample of the reference asset by applying the coefficient set B to equation (1) and adding the residual w_T.

$$\beta_1 \cdot z_{a1,l1} + \beta_2 \cdot z_{a2,l2} + ... + \beta_m \cdot z_{am,lm} + ... + \beta_M \cdot z_{aM,lM} = \widehat{y}_T + w_T \tag{13}$$

The difference of the estimated return derived from the GA optimization $\widehat{y}_T$ and the real world return value y_T is then:

$$y_T - \widehat{y}_T = w_T + u_T \tag{14}$$

By carrying out this optimization repeatedly (with a repetition factor R) and calculating the mean value of these estimated returns we finally get:

$$y_T - \frac{1}{R}\sum_{r=1}^{R} \widehat{y}_T(r) \approx u_T \qquad \Rightarrow \qquad \widehat{y}_T = \frac{1}{R}\sum_{r=1}^{R} \widehat{y}_T(r) \tag{15}$$

During the GA optimization, we monitor the optimization error defined in (6) rather than the residual vector W. Therefore we need a corresponding scalar that reflects the noise vector U. To do so, we first calculate the noise vector U of the LS model. From that, we compute the geometrical mean of the elements u_i of vector $\widehat{U}$ with length N and eventually gain the *noise value* $\overline{u}$:

$$\overline{u} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} u_i^2} \qquad with \quad \widehat{U} = Y - \widehat{B} \cdot Z = \{u_i\} \tag{16}$$

According to above theory, we may derive $\widehat{y}_T$ when repetitively performing GA optimizations solving (10), with each single optimization being aborted at an optimization error (6) matching $\overline{u}$.

4 Implementation

The GA model may be described as follows:

- Population: Set of chromosomes, each representing a possible weighting coefficient vector B.
- Initial population: The initial size of the weighting coefficients are assumed to be of approximately the same magnitude, with the sum of all coefficients to be one. Therefore coefficients are initially set by a random number with standard deviation std_{init} = 1 / number of coefficients per chromosome.
- Mutation: Add a normal distributed random value with given standard deviation std_{GA} to a coefficients of a chromosome with probability p_m.
- Crossover: Crossover of two randomly selected (probability p_c) chromosomes with randomly selected crossover point. The Offspring replace their parents.
- Fitness calculation: The Fitness function is implemented according to equation (6).
- New population: New populations are selected using the fitness proportionate roulette wheel selection method.
- Freezing: To keep the values of the weighting coefficients in a realistic range and to speed up the optimization process, the updating value of the mutation operator is frozen whilst operation by repeatedly reducing the standard deviation factor std_{GA}. If freezing is not established, some weighting coefficients tend to adapt to unrealistic large numbers.[4]
- Optimization cycles: The GA optimization shall be terminated when the optimization error equals the noise value $\overline{u}$. This is done indirectly by predefining the number of optimization cycles. This simple method to control the optimization error proved to be very accurate. Moreover, it allows for a simple mechanism to implement the freezing mechanism. Freezing of std_{GA} is performed at a fraction of the total number of optimization cycles.
- Repetition: The whole GA optimization is performed R times with the final estimation computed according to (15).

[4] This is due to the fact, that we optimize an unrealistic model which neglects U.

5 Numerical Results

5.1 Measurement of the Estimation Quality

Measuring the quality of the estimation will be conducted by actively managing the IFX stock according to the daily return forecast. In case of an estimated yield which is positive, we will either buy the share (in the case that we do not own it) or keep it (when it has been already bought before). In case of an estimated yield which is negative, we will either sell it (when we own the stock) or do nothing (when the stock has been sold before). The return of the managed asset is the return we gain after 100 trading days of applying this strategy. The annualized ROI might be gained by extrapolating the return results to a whole year by simply multiplying managed asset return by a factor of 2.6, a virtual number though, which should be judged with caution. The first observed trading day is January the 26th of 2012 and the last observed day is June the 13th in 2012. Note that no transaction costs have been considered in this measurement. The standard deviation of the IFX stock return within this period is 0.024. Within this period the IFX stock fell from EUR $7,02$ to EUR $6,16$.

5.2 Settings of the Genetic Algorithm

The parameters of the GA have been chosen empirically, such that an approximately constant number of improvements per iteration is established. The standard deviation for the initial population generation and the first value of standard deviation value for the mutation operand have been chosen to reflect a typical value of weighting coefficients. Consecutive values of std_{GA} have been selected to establish an approximately constant number of improvements during the optimization cycles.

- population size: 100 chromosomes.
- Probability of crossover: $p_c = 0.01$.
- Probability of mutation: $p_m = 0.01$.
- Standard deviation for the initial population generation: $std_{init} = 0.1$.
- Standard deviation value for the mutation operand: $std_{GA} = (0.1, 0.05, 0.03, 0.01)$. These 4 values are implemented one after another, at a quarter of the total optimization cycle number respectively to perform freezing.
- Number of optimization cycles: default setting: 200 optimization cycles referring to an optimization error of 0.12.
- Number of lags M (default setting): 20.
- Threshold value (default setting): as given by equation (3).
- Repetition factor R (number of repeated optimizations): 50 or 100.
- Window size T_{ccf} for the ccf to perform lag selection: 50 samples.
- Lag number restricted to $lm < 20$.

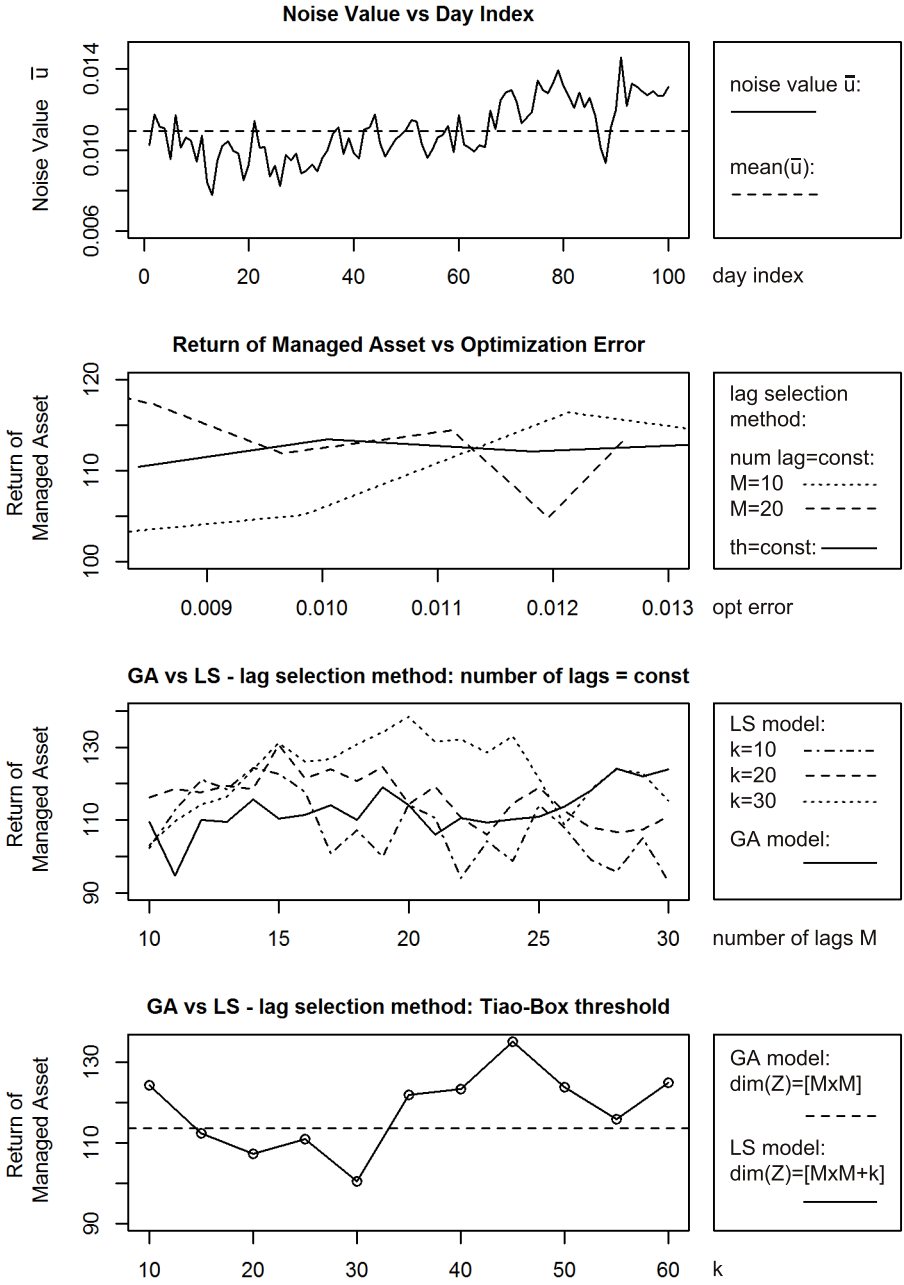


Fig. 1. Simulation Results

5.3 Simulation Results

Figure (1) shows numerical results we obtained from the simulations.

Before running the GA optimization, the magnitude of the noise value is required to calibrate the GA optimization. Therefore the noise value is obtained by doing a LS simulation, and will be used as a directive to set the GA model optimization error subsequently. Part 1 of Figure 1 shows the noise value, which has been measured at every trading day within the defined trading period of 100 days. The noise value varies from 0.008 to 0.013 with a mean of approximately 0.011.

Part 2 depicts the return of the managed asset using forecasts from the GA model. The GA optimizations have been performed with different optimization error settings, reflecting the spread of the noise value as shown in Part 1. This evaluation is performed by lag selection methods (as described in Section 2.2, i.e. by applying a constant number of lags M (10 and 20)) as well as by setting a constant threshold value according to Tiao and Box defined in (3). Part 2 does not show any significant curve progression and variations of the return at different optimization errors may rather reflect the statistical behavior of the simulation than highlighting the optimization error to be optimal.

Part 3 compares the obtained return of a managed asset (y-axis) from the GA based VAR model versus the LS based VAR model when applying the constant lag number method. Here, a predefined number of strongest lags (number of lags M, horizontal axis) are selected per ccf. Results of the GA model (solid line) show a rather flat line with a slight increase as M is increased if we ignore low returns for small values of $M \leq 12$. The LS method has been tested for different values of k, which expresses how much the system of equations for the LS optimization is overdetermined. As expected, larger k yields better returns. Assuming a VAR model with $M = 15$, the total number of past samples required for the GA model would typically add up to $L_{GA} = 25$ (with $max(lm) \approx 10$). Compared to the best LS model with $k = 30$, the total number of past samples used for coefficient estimation is $LLS = 45$, and hence more than two months[5] are required for this LS model to settle from an exogenous shock.

The fourth part provides a comparison of the GA and LS models when applying the constant threshold lag selection method. The LS model (solid line) has been tested for different values of k (x-axis) and is compared to the GA model (dotted line).[6] The number of selected lags is defined inherently by the threshold value (3) and therefore varies from day to day. On average M is about 50 and its maximal value exceeds 90. Lowering the threshold to lower M would not work, since M would soon become zero for some trading days. Therefore the constant threshold method can't be recommended for modeling a robust estimation model.

[5] 45 trading days are approx. 63 calendar days.

[6] Since k is not applicable to the GA method, the according graph is a straight line.

An LS model based computation of estimates for 100 trading days required less than a minute on a standard PC. The GA model, however, required approximately 30 hours to compute 100 estimation with 50 repetitions each. Parallelization is necessary to include the method into real-world environments.

6 Conclusion

We have shown that a GA based model constitutes comparable estimation performance than a state-of-the-art LS model. This was proved numerically by using return forecasts to actively manage the Infineon stock and compare the return of the managed asset of the GA versus the LS method after 100 trading days. The GA optimization method can be designed to have a much shorter recovery time in the case of exogenous shocks. This is due to the shorter sample history which is required for the GA based coefficient estimation. A drawback of the GA model is its larger simulation time compared to the LS model. Simulations have been run for a single stock and for a single time period. Hence more simulations are necessary to improve the credibility.

References

1. Agapitos, A., O'Neill, M., Brabazon, A.: Evolving Seasonal Forecasting Models with Genetic Programming in the Context of Pricing Weather-Derivatives. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 135–144. Springer, Heidelberg (2012)
2. Azzini, A., Dragoni, M., Tettamanzi, A.G.B.: Using Evolutionary Neural Networks to Test the Influence of the Choice of Numeraire on Financial Time Series Modeling. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Drechsler, R., Farooq, M., Grahl, J., Greenfield, G., Prins, C., Romero, J., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Urquhart, N., Uyar, A.Ş. (eds.) EvoApplications 2011, Part II. LNCS, vol. 6625, pp. 81–90. Springer, Heidelberg (2011)
3. Brabazon, A., O'Neill, M.: Biologically Inspired Algorithms for Financial Modelling. Springer (2006)
4. Chung, F., Fu, T., Ng, V., Luk, R.: An evolutionary approach to pattern-based time series segmentation. IEEE Transactions on Evolutionary Computation 8(5), 471–489 (2004)
5. Dang, J., Brabazon, A., O'Neill, M., Edelman, D.: Estimation of an Egarchvolatility Option Pricing Model using a Bacteria Foraging Optimisation Algorithm. In: Brabazon, A., O'Neill, M. (eds.) Natural Computing in Computational Finance. SCI, vol. 100, pp. 109–127. Springer, Heidelberg (2008)
6. Ferreira, T., Vasconcelos, G., Adeodato, P.: A new evolutionary method for time series forecasting. In: ACM GECCO 2005, pp. 2221–2222 (2005)

7. Larkin, F., Ryan, C.: Modesty Is the Best Policy: Automatic Discovery of Viable Forecasting Goals in Financial Data. 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, Part II. LNCS, vol. 6025, pp. 202–211. Springer, Heidelberg (2010)
8. Lütkepohl, H.: New Introduction to Multiple Time Series Analysis. Springer (2007)
9. Sims, C.: Macroeconomics and reality. Econometrica 48(1), 1–48 (1980)
10. Tsay, R.: Analysis of Financial Time Series. Wiley Series in Probability and Statistics. John Wiley & Sons (2005)
11. Tuite, C., Agapitos, A., O'Neill, M., Brabazon, A.: A Preliminary Investigation of Overfitting in Evolutionary Driven Model Induction: Implications for Financial Modelling. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Drechsler, R., Farooq, M., Grahl, J., Greenfield, G., Prins, C., Romero, J., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Urquhart, N., Uyar, A.Ş. (eds.) EvoApplications 2011, Part II. LNCS, vol. 6625, pp. 120–130. Springer, Heidelberg (2011)

Usage Patterns of Trading Rules in Stock Market Trading Strategies Optimized with Evolutionary Methods

Krzysztof Michalak[1], Patryk Filipiak[2], and Piotr Lipinski[2]

[1] Institute of Business Informatics,
Wroclaw University of Economics, Wroclaw, Poland
krzysztof.michalak@ue.wroc.pl
[2] Institute of Computer Science,
University of Wroclaw, Wroclaw, Poland
{patryk.filipiak,lipinski}@ii.uni.wroc.pl

Abstract. This paper proposes an approach to analysis of usage patterns of trading rules in stock market trading strategies. Analyzed strategies generate trading decisions based on signals produced by trading rules. Weighted sets of trading rules are used with parameters optimized using evolutionary algorithms. A novel approach to trading rule pattern discovery, inspired by association rule mining methods, is proposed. In the experiments, patterns consisting of up to 5 trading rules were discovered which appear in no less than 50% of trading experts optimized by evolutonary algorithm.

Keywords: stock market, trading rules, evolutionary computing.

1 Introduction

Evolutionary computing is often applied to various economic and financial problems [1,7,16] such as optimization of investment portfolios [3,13,15,18] and supporting financial decision making [9,12,22]. Training of trading experts based on sets of trading rules is an important task with practical applications in stock market trading [6,10,11].

In this paper we consider optimization of trading experts based on a predefined set of trading rules. An individual rule may be parameterized by several variables such as time period for which underlying indicators are calculated and various coefficients. Most often "buy" and "sell" signals are generated in response to certain indicators crossing each other or crossing a selected threshold. In the latter case, thresholds used by the rule are also considered as parameters affecting the behaviour of this particular rule. Each expert is based on the same set of rules, but with differing weights from range $[-1, 1]$, so, in principle, it may skip certain rules (weight $= 0$) or even negate their behaviour (weight < 1). Final decision returned by an expert at any time instant is determined by calculating a weighted sum of outputs of all trading rules and by applying buy and sell

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 234–243, 2013.
© Springer-Verlag Berlin Heidelberg 2013

decision thresholds assigned to that particular expert. Expert parameters form a vector with numeric coordinates each of which is interpreted accordingly as: rule weight, rule parameter (time period, coefficient, rule decision threshold) or expert decision threshold.

The approach used in this paper consists of application of two separate steps: evolutionary optimization that adjusts parameters of trading experts in order to achieve good performance and frequent set analysis algorithm which identifies patterns of trading rules that occur together in successful experts. Identification of frequent patterns in trading rules is intended as a further step in relation to work described in [14]. The first step, i.e. expert parameters optimization was performed using evolutionary algorithm described later in the paper which ran for a predefined number of generations. The performance of the trading expert was optimized separately for each trading day (with the assumption, that only intra-day trading is performed) and a separate optimization procedure was performed for each stock. From the optimized population a constant fraction of best experts was taken as input data for the process of discovering patterns of usage of trading rules.

2 Evolutionary Optimization

Trading experts generate decisions based on trading rules. The rules require various parameters, the signals generated by the rules are weighted and compared to decision thresholds. Therefore, the behaviour of a trading expert can be parameterized by a vector of constant length with coordinates corresponding to the above mentioned parameters.

Vector parameterizing a trading expert can be used as a chromosome of a specimen in an evolutionary algorithm. Such a chromosome can be stored in the following form:

$$weight_1, par_{1,1}, \ldots, par_{1,k_1}, \ldots, weight_n, par_{n,1}, \ldots, par_{n,k_n}, \Theta_{buy}, \Theta_{sell}$$

where:
$weight_i$ - i-th rule weight,
$par_{i,j}$ - j-th parameter of the i-th rule,
k_i - number of parameters for the i-th rule,
Θ_{buy}, Θ_{sell} - decision thresholds.

In order to optimize the performance of trading experts an evolutionary algorithm Algorithm 1 was employed.

In the presented algorithm the following operations are used:

SelectMatingPool - selection of a mating pool for crossover and mutation consisting of N_{mate} specimens. This selection is performed using a series of binary tournaments in which the better one of the two randomly selected specimens is included in the mating pool.

Crossover - a single-point crossover operator.

Algorithm 1. Evolutionary algorithm used for optimizing trading experts

N_{pop} - population size
N_{mate} - mating pool size
P_{mut} - probability of mutating a single parameter

$P_1 = \text{InitPopulation}()$
$\text{Evaluate}(P_1)$
for $i = 1 \rightarrow N_{Gen}$ **do**
  $P'_i = \text{SelectMatingPool}(P_i, N_{mate})$
  $P'_i = \text{Crossover}(P'_i)$
  $P'_i = \text{Mutate}(P'_i, P_{mut})$
  $\text{Evaluate}(P'_i)$
  $P_i = P_i \cup P'_i$
  $P_{i+1} = \text{Reduce}(P_i, N_{pop})$
end for

Mutate - a mutation procedure which changes any given vector coordinate with P_{mut} probability.

Evaluate - specimen evaluation based on simulated performance of the trading expert. For a given specimen trading decisions are generated by an expert based on parameters vector equal to the genotype of the specimen. The evaluation of the specimen is the return which would be achieved if the suggestions of the expert were used to buy and sell stocks in the intra-day trading model. Additionally, a specimen is required to perform at least one transaction during the day (specimens that do not generate any "buy" signals are therefore given an evaluation of 0). A commision of 0.4% is applied to each transaction (each buy and each sell).

Reduce - reduction of $P_i \cup P'_i$ back to N_{pop} specimens with N_{elite} best specimens guaranteed to be copied to the next population and the rest filled using fitness proportionate selection.

Parameters of the evolutionary algorithm in the experiments were set as presented in Table 1.

Table 1. Parameters of the evolutionary algorithm

Desctiption	Symbol	Value
Number of generations	N_{gen}	30
Size of the population	N_{pop}	100
Size of the mating pool	N_{mate}	50
Number of the best specimens copied to the next population	N_{elite}	20
Probability of mutating each parameter	P_{mut}	0.1

3 Analysis of Trading Rule Sets

In this paper we aimed at finding sets of trading rules that appear together in trading experts optimized using evoluationary algorithm. The data set for pattern discovery was a top 20% of specimens present in the last population in evolutionary algorithm (after N_{gen} generations). These 20% of specimens were selected separately for each stock and each trading day. To increase the size of the data set (which is only 20 specimens for $N_{pop} = 100$) specimens for each stock obtained on all trading days were used together in a single set for pattern discovery.

The general approach is similar to frequent set discovery techniques used in association rule mining [4,5,17]. The algorithm that generates sets of trading rules that are frequently found in good trading experts is presented in Algorithm 2. It builds frequent rule sets incrementally, starting from sets containg one element each (which are simply equivalent to individual rules). Building larger sets is based on the observation, that if a set is frequent then all of its subsets must also be frequent. Based on this property it is possible to build larger frequent sets by extending the already found smaller frequent sets. The support needs to be calculated for the newly constructed sets, but there is a gain in that the number of candidate sets is lower than when all possible sets of given size would be considered. This algorithm is parameterized by the number *minSupport* which defines what percentage of specimens a rule set must appear in to be considered frequent. In the experiments this number was set to $minSupport = 50\%$. The rule is considered to be used in a particular expert if the absolute value of its weight is not less than $minWeight$ parameter. In the experiments the minimum weight was set to $minWeight = 0.7$.

Algorithm 2. Frequent rule set discovery algorithm

```
i = 1
C_1 = family of sets, each containing one rule
S_1 = CheckSupport(C_1)
while |S_i| > 1 do
  C_{i+1} = Combine(S_i)
  S_{i+1} = CheckSupport(C_{i+1})
  i = i + 1
end while
```

The procedures used in this algorithm are as follows:

CheckSupport - processes a family C_i of candidate sets of rules. For each set of rules $R \in C_i$ iterates over the entire data set V (consisting of N specimens). If in a given specimen each of the rules in the candidate set R has a weight with absolute value at least $minWeight$ the count m is increased. The procedure returns those candidate sets from C_i, which have $m \geq minSupport * N$. The CheckSupport procedure is detailed in Algorithm 3.

Algorithm 3. CheckSupport - a procedure for selecting rule sets with support $\geq minSupport$

C_i - a family of candidate sets
S_i - a family of frequent sets
V - data set of the best specimens generated by evolutionary algorithm
N - the size of the data set V ($N = |V|$)

$S_i = \emptyset$
for $R \in C_i$ **do**
  $m = 0$
  for $v \in V$ **do**
    if $\forall k \in R \cdot |v[k]| \geq minWeight$ **then**
      $m = m + 1$
    end if
  end for
  if $m \geq minSupport * N$ **then**
    $S_i = S_i \cup \{R\}$
  end if
end for

Combine - given a family of frequent sets S_i of size i generates a family of candidate sets C_{i+1} of size $i + 1$. This family includes all sets that have the property, that each subset of size i of each candidate set is in S_i (i.e. is a frequent set). The Combine procedure is detailed in Algorithm 4.

Algorithm 4. Combine - a procedure generating a family C_{i+1} of candidate sets of size $i + 1$ based on a family S_i of frequent sets of size i

C_{i+1} - a family of candidate sets of size $i + 1$
S_i - a family of frequent sets of size i

$C_{i+1} = \emptyset$
for $T_j \in S_i$ **do**
  for $T_k \in S_i$ **do**
    if $|T_j \backslash T_k| = 1$ **and** $|T_k \backslash T_j| = 1$ **then**
      $C_{i+1} = C_{i+1} \cup \{T_J \cup T_k\}$
    end if
  end for
end for

4 Experiments

The experiments were performed on high-frequency data from the New-York Stock Exchange. 50 stocks were analyzed: AAPL, AMZN, BIDU, BIIB, C, CAT, CF, CHK, CMI, CRM, CSCO, CVX, DIA, EMC, EOG, F, FAS, FFIV, GE, GLD, GLW, GOOG, GS, HPQ, IBM, INTC, IVV, JPM, MDY, MO, MS, MSFT,

Table 2. Maximum sizes of frequent rule sets and the number of sets of the maximum size

Ticker	Max set size	Num. sets	Ticker	Max set size	Num. sets	Ticker	Max set size	Num. sets
AAPL	4	471	FFIV	3	235	NVDA	3	2907
AMZN	3	1	GE	3	14	ORCL	3	4237
BIDU	3	141	GLD	3	2057	PFE	3	1622
BIIB	3	1466	GLW	5	7	PX	2	1267
C	3	249	GOOG	4	2	SPG	3	86
CAT	3	191	GS	3	518	SPY	3	1456
CF	3	724	HPQ	3	1310	T	5	29
CHK	3	288	IBM	3	567	TLT	3	277
CMI	4	4	INTC	3	197	UNP	3	1
CRM	3	105	IVV	3	15	V	3	277
CSCO	3	227	JPM	3	920	VALE	3	1190
CVX	3	74	MDY	3	4227	VZ	3	569
DIA	2	2177	MO	3	3	WFC	2	899
EMC	3	2748	MS	3	1777	WYNN	5	14
EOG	3	9	MSFT	3	118	XLF	3	196
F	3	192	NFLX	5	1	YHOO	4	5
FAS	4	5	NKE	2	1725			

Table 3. Examples of frequent rule sets of size 5 obtained in the experiments

Ticker	Rules		
GLW	$EMV_{(3)}$+TBR, LinDir,	$EMV_{(3)}$+Trend, $EMV_{(3)}$+NormPrice	$EMV_{(3)}$+WiO
NFLX	$EMV_{(2)}$+RoC, $EMV_{(2)}$+NormPrice,	K-Stochastic, RSI	NormPrice+WiO
T	NormPrice+MA, NormPrice+$V\&P_{(1)}$,	NormPrice+two MAs, $V\&P_{(2)}$	NormPrice+TBR,
WYNN	$EMV_{(1)}$+OvO, RoC,	$EMV_{(2)}$+$V\&P_{(2)}$, WiO	NormPrice+two MAs,

NFLX, NKE, NVDA, ORCL, PFE, PX, SPG, SPY, T, TLT, UNP, V, VALE, VZ, WFC, WYNN, XLF and YHOO. Minute quotations were used from trading days in the period from 2012.01.01 to 2012.06.30. This period contains 125 trading days.

A set of 78 trading rules was used. The rules were based on indicators such as Moving Averages [21], Chaikin Oscillator [19], Overbought/Oversold [20], RSI [2], Williams %R Oscillator [8], etc. Due to space limitations it is not possible to describe all the rules in detail. Some of the indicators were based on moving

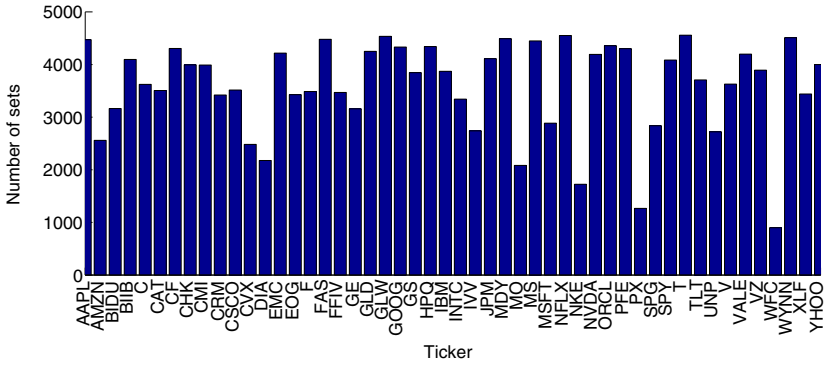


Fig. 1. Number of 2-element frequent rule sets for stocks used in the experiments

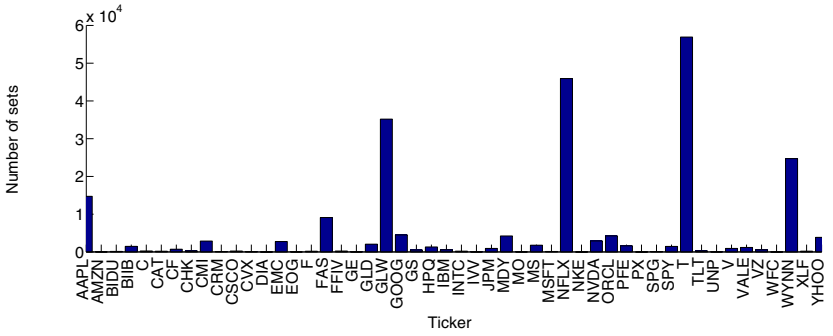


Fig. 2. Number of 3-element frequent rule sets for stocks used in the experiments

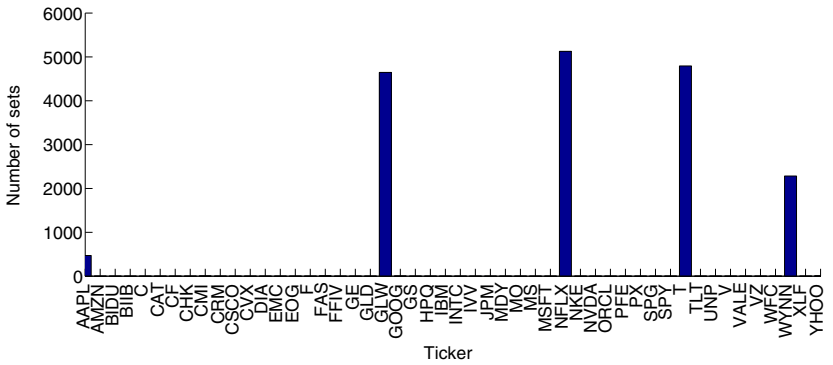


Fig. 3. Number of 4-element frequent rule sets for stocks used in the experiments

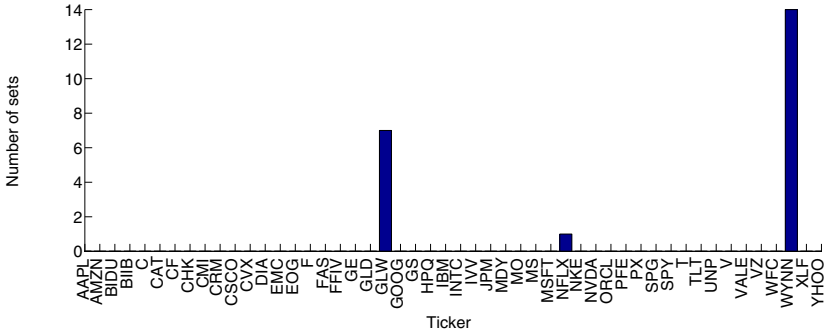


Fig. 4. Number of 5-element frequent rule sets for stocks used in the experiments

averages. These indicators were used in conjunction with both the simple moving average (SMA) and the exponential moving average (EMA). Thus, the total number of rules active in the system was 96.

The rules were further parameterized. For example if moving average was used, the period of the moving average was modified by the evolutionary algorithm. Periods were adjusted in the range $[5, 30]$. Other parameters and decision thresholds were adjusted within ranges depending of the meaning of the parameter.

The results of experiments are summarized in Table 2. This table shows the maximum size of frequent rule set obtained for each stock and the number of frequent rule sets of the maximum size generated in the experiments. The maximum size of frequent rule sets varies from 2 to 5 depending on what stock the rules are related to. For most of the stocks frequent rule sets were found containing at least 3 rules. In the experiments no frequent rule sets containing 6 or more rules were found. While such sets were generated as candidate sets, none of them has reached the required support of at least 50% (however, some of the 6-elements sets have reached the support of about 47.5%).

The number of n-element frequent rule sets found in the experiments (for n=2,3,4 and 5) is shown in Figures 1-4.

Frequent rule sets of larger size seem to be more interesting because they may indicate more complex dependencies among the rules. The maximum frequent rule set size obtained in the experiments was 5. In Table 3 examples of frequent rule sets of size 5 are presented.

Abbreaviations used in Table 3 have the following meaning.

EMV - Ease of Movement Value
LinDir - Linear Regression Direction
NormPrice - Price normalized by Dow Jones Index
OvO - Overbought / Oversold
RoC - Rate of Change
RSI - Relative Strength Index
TBR - Top / Bottom Reversal
V&P - Volume and Price
WiO - Williams Oscillator

5 Conclusion

In this paper a task of finding repetitive patterns in trading experts was approached. Trading experts were constructed from trading rules. Parameters and weights of the trading rules were optimized using evolutionary algorithm in order to improve performance of the experts. Based on the optimized experts a discovery of usage patterns of the trading rules was performed.

The novel approach to this topic proposed in this paper is inspired by frequent set discovery methods used in assiociation rule mining. These methods were adapted to finding patterns emerging in a population of trading experts optimized by an evolutionary algorithm to achieve good performance in stock market trading. The results of experiments have shown that patterns of up to 5 trading rules emerge in the population. Each of the patterns is supported in at least 50% of the optimized experts.

The results of the experiments suggest, that repetitive patterns are present in trading rule sets. In the context of evolutionary optimization of trading rules such patterns could be used to improve solutions in dynamically changing environment of stock market trading, especially for shortening generation of good populations for evolutionary algorithms.

References

1. Bauer, R.: Genetic Algorithms and Investment Strategies. Wiley, Chichester (1994)
2. Bauer, R.J., Dahlquist, J.R.: Technical Markets Indicators: Analysis & Performance, p. 129. John Wiley & Sons (1998)
3. Best, M.J.: Portfolio Optimization. Chapman&Hall/CRC (2010)
4. Borgelt, C.: Frequent item set mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(6), 437–456 (2012)
5. Chopra, D., Vishwakarma, D.: Efficient Frequent Item set Discovery Technique in Uncertain Data. International Journal of Engineering and Advanced Technology (IJEAT) 1(6) (2012)
6. Dempsey, I., O'Neill, M., Brabazon, A.: Adaptive Trading with Grammatical Evolution. In: Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006), pp. 2587–2592. IEEE, Los Alamitos (2006)
7. Dempster, M., Jones, C.: A Real-Time Adaptive Trading System using Genetic Programming. Quantitative Finance 1, 397–413 (2001)
8. Kirkpatrick, C.D., Dahlquist, J.R.: Technical Analysis: The Complete Resource for Financial Market Technicians, pp. 440–441. FT Press (2010)
9. Li, J., Taiwo, S.: Enhancing Financial Decision Making Using Multi-Objective Financial Genetic Programming. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2171–2178 (2006)
10. Lipinski, P.: Dependency Mining in Large Sets of Stock Market Trading Rules. In: Pejas, J., Piegat, A. (eds.) Enhanced Methods in Computer Security, Biometric and Intelligent Systems, pp. 329–336. Kluwer Academic Publishers (2005)
11. Lipinski, P.: Discovering Stock Market Trading Rules Using Multi-layer Perceptrons. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 1114–1121. Springer, Heidelberg (2007)

12. Lipinski, P.: ECGA vs. BOA in Discoverying Stock Market Trading Experts. In: Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2007, pp. 531–538. ACM (2007)
13. Lipinski, P.: Evolutionary Strategies for Building Risk-Optimal Portfolios. In: Brabazon, A., O'Neill, M. (eds.) Natural Computing in Computational Finance. SCI, vol. 100, pp. 53–65. Springer, Heidelberg (2008)
14. Lipinski, P.: Frequent Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis. In: Brabazon, A., O'Neill, M., Maringer, D.G. (eds.) Natural Computing in Computational Finance. SCI, vol. 293, pp. 131–145. Springer, Heidelberg (2010)
15. Michalak, K., Filipiak, P., Lipiński, P.: Evolutionary Approach to Multiobjective Optimization of Portfolios That Reflect the Behaviour of Investment Funds. In: Ramsay, A., Agre, G. (eds.) AIMSA 2012. LNCS, vol. 7557, pp. 202–211. Springer, Heidelberg (2012)
16. Michalak, K., Lipinski, P.: Prediction of high increases in stock prices using neural networks. Neural Network World 15(4), 359–366 (2005)
17. Park, B.J.: Efficient Tree-based Discovery of Frequent Itemsets. International Journal of Multimedia and Ubiquitous Engineering 7(2) (2012)
18. Radziukyniene, I., Zilinskas, A.: Evolutionary Methods for Multi-Objective Portfolio Optimization. In: Proceedings of the World Congress on Engineering 2008, pp. 1155–1159. Newswood Limited (2008)
19. Ranganatham, M.: Investment Analysis and Portfolio Management, pp. 387–388. Pearson Education (2004)
20. Schwager, J.D.: Technical Analysis. pp. 174–178. John Wiley & Sons (1995)
21. Srivastava, U.K., Shenoy, G.V., Sharma, S.C.: Quantitative Techniques For Managerial Decisions, pp. 392–394. New Age International (1989)
22. Tsang, E., Li, J., Markose, S., Er, H., Salhi, A., Iori, G.: EDDIE in Financial Decision Making. Journal of Management and Economics 4(4) (2000)

Combining Technical Analysis and Grammatical Evolution in a Trading System

Iván Contreras[1], J. Ignacio Hidalgo[1], and Laura Núñez-Letamendia[2]

[1] Facultad de Informática, Universidad Complutense de Madrid, - Spain
ivancontreras@pas.ucm.es,hidalgo@dacya.ucm.es
[2] IE Business School, Madrid - Spain
Laura.Nunez@ie.edu

Abstract. Trading Systems are beneficial for financial investments due to the complexity of nowadays markets. On one hand, finance markets are influenced by a great amount of factors of different sources such as government policies, natural disasters, international trade, political factors etc. On the other hand, traders, brokers or practitioners in general could be affected by human emotions, so their behaviour in the stock market becomes nonobjective. The high pressure induced by handling a large volume of money is the main reason of the so-called market psychology. Trading systems are able to avoid a great amount of these factors, allowing investors to abstract the complex flow of information and the emotions related to the investments. In this paper we compare two trading systems based on Evolutionary Computation. The first is a GA-based one and was already proposed and tested with data from 2006. The second one is a grammatical evolution approach which uses a new evaluation method. Experimental results show that the later outperforms the GA approach with a set of selected companies of the spanish market with 2012 data.

1 Introduction

The complexity of the search space in the problem of finding the optimal series of investments makes metaheuristics one of the best ways to achieve a good solution in short time. Therefore, this problem becomes a challenge to be faced by researchers. The field of metaheuristics applied to trading problems has been growing rapidly, both within the scientific and the professional world. As a consequence investment systems have benefited from the development of complex computer-aided systems and the large amount of data and information available (some of them using evolutionary metaheuristics).

We can find in the literature different approaches using Evolutionary Algorithms (EAs) for discovering Trading Rules. For instance, Allen and Karjalaein [3] proposed to use Genetic Algorithms (GAs) to find technical Trading Rules. Nuñez [17] designed several GA models to develop financial investment strategies. This work was applied on the 1987-1996 share price data from the Madrid Stock Exchange (Spain) and we can not affirm that conclusions can be extrapolated to nowadays economic circumstances. In [14] the authors describe a GA-based trading system that uses different kinds of rules for market and companies

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 244–253, 2013.
© Springer-Verlag Berlin Heidelberg 2013

information. The system is applied, on a daily basis, to companies belonging to the S&P 500 index. In this work they propose to apply the methodology to Technical Analysis (TA) and fundamental Analysis (FA) separately and the authors claim that the main problem is the computational time required for training the trading system with the daily data of stock prices. This restriction was partially solved in [8] [9] with the implementation of a GA over a parallel computer architecture. There are also other approaches that use Genetic Programming (GP) to obtain a set of rules and signals for investing [15] [16]. More recently, Bodas et al. in [4] and [22] showed an approximation using a multi-objetive approach and optimizing the parameters of the selected Technical Indicators every time a new data was read. This approach improves not only performance but also profits. However they used only two Technical Indicators (TI).

A different approach was presented in [7]. In this work, the authors proposed the use of EAs to optimise jointly the parameters of selected FA and TA Indicators. Although experimental results seemed to be satisfactory, it was concluded that it would be necessary to test this methodology with new data. The current economic circumstances do not appear to be the same as those simulated in the aforementioned work.

Grammatical Evolution (GE) is a relatively new evolutionary alternative to the above mentioned proposals [19]. GE is presented as a smart solution that fits perfectly in the context of the complexity of Stock Markets. Thus, other previous works already made use of GE as a methodology to invest in stock markets. The BDS Group at the University of Limerick has carried out an excellent work about this topic, for example in [10] the authors propose a GE to evolve a financial trading system. In this approach the authors show an adaptive grammar with a variant of the moving window. The different rules of the grammar are in constant evolution during the execution of the trading system while new data is uploading. Other works of this group show the proficiency of GE in foreign-exchange markets or in different indexes of the stock market as [5] [11] among others. Other authors proposed trading system based in GE, as in [1], where the authors build a system that uses the co-evolution of the entries, exits and stop loss for long positions, and short positions. These authors update their work in [2] where they change the fitness function (sharpe index) with a complex fitness proposed by P.Saks [20]. Those works inspired our GE implementation. We have changed the way the solutions are evaluated. The main target of this work is to analyse the results provided by two trading systems within a recession economy period and to propose a methodology that works properly in a hostile environment. In this paper we set the initial stage for the development of our next project that will allow an exhaustive analysis of the more profitable companies in major world regions. So, the contributions of the paper are two-fold:

1. The Trading System presented in [7] is tested with new data from 2012. The results are not as satisfactory as those obtained with the 2006 data. Thus we check that earnings depend on the environment and selected period.
2. We need a new approach able to obtain profits within a recession period. We propose the application of a new evolutionary technique based on grammars.

Experimental results with this proposal are encouraging, both for the data used in the previous articles, as well as for the new data.

The rest of the paper is organized as follows. Section 2 briefly describes the GA Trading System (TS) previously presented in [7]. Section 3 presents the methodology which uses GE to obtain the trading rules. Section 4 presents the experimental results when applying the proposed methodology to a set of companies during a period of 2012 and we compare them with those of the GA of [7]. Finally, we conclude the paper and outline some future research lines in Section 5.

2 GA Based Trading System

The details of the GA system were presented in [7]. However we will discuss briefly its main characteristics. In the mentioned paper the authors propose the use of an EA to optimise the trading system. The objective was to obtain a set of trading signals indicating *buy*, *sell* or do nothing. A solution gives a set of values indicating weights and parameters or thresholds for a pre-selected set of technical and fundamental indicators. The evolutionary approach uses a GA with a Filling Operator (GAwFO), which is basically a simple GA with a modification of the selection and crossover operators. The trading system works as follows:

1. The investor selects a set of Technical Indicators for TA
2. The investor selects a set of Fundamental Indicators for FA
3. Establish $Threshold_{buy}$ and $Threshold_{sell}$ ranges and $Weights$ ranges
4. For each company i
   (a) Apply GAwFO over a period of years to obtain a solution
   (b) Apply TA and FA using the parameters given by (a) to the target year
      i. For each indicator I_j generate the indicator signal $I_j s$ (Buy = 1, Sell = -1 or Neutral = 0) as follows:
         A. if $I_j > or < Threshold_{buy}$ on a day Buy $I_j s = 1$
         B. elseif $I_j > or < Threshold_{sell}$ on a day Sell $I_j s$ =-1
         C. else Neutral $I_j s$=0
      ii. Compute the Raw Trading System Signal by adding the indicators signals weighted by their weights $RTS_s = \sum_{j=1}^{n} I_j s \cdot W_j$
      iii. Compute the Net Trading System Signal choosing values for X and Y as follows:
         A. If $RTS_s = or > X$ then $TS_s = 1$
         B. elseif $RTS_s = or < -Y$ then $TS_s = -1$
         C. else $TS_s = 0$
      iv. Compute the profit given by the Trading System

The TS presented in that paper combines TA and FA. TA is formulated by four Technical Indicators (TI); Moving Average (MA), market Volume (V), Relative Strength Index Divergences (RSI) and Support and Resistances (SR). FA also uses four Fundamental Indicators (FIs): Price Earning Ratio (PER), Price Book Value (PBV), Return On Assets (ROA) and Sales Growth (SG). Each one of

these indicators gives us a signal of buy, sell or neutral. There are eight weights corresponding to each indicator that weight the importance of each indicator in obtaining buy or sell signals. An individual of the GA encodes the thresholds and the weights to be applied.

3 GE Based Trading System

A relatively new approach called Grammatical Evolution (GE) arose during the last years as a powerful EA tool. This evolutionary computation technique was promoted by C.Ryan, JJ. Collins and M. O'Neill in 1998 [19]. We can summarise the definition of GE as a type of Evolutionary Algorithm designed to evolve computer programs defined by a grammar (usually in BNF notation). The most similar procedure is Genetic Programing (GP), which is able to evolve computers programs as well. The main dissimilarity that makes the GE an attractive and elegant solution is the difference between the phenotype and the genotype. Thus, instead of representing the programs as a tree-solution, GE presents a chromosome composed by codons that are directly connected with a specific rule of the grammar. The chromosome itself is considered the genotype and the real code derivative of the codons is called phenotype.

3.1 Fitness Fuction

The fitness function which guides our trading system is the accumulated return of the investments for the analysed period. Each one of the investments are guided by the signals that the trading algorithm provides (buy, sell or nothing).

$$AR_i = \prod_{f}^{i=1}(1 + DR_i) \tag{1}$$

Where AR_f is the accumulated return at the end of the trading period and DRi is the daily return given by:

$$DR_i = \begin{cases} \frac{P_i - P_{i-1}}{P_{i-1}} & \text{if the TS gives a long signal} \\ \frac{-(P_i - P_{i-1})}{P_{i-1}} & \text{if the TS signal is short selling} \\ \emptyset & \text{if the TS signal is neutral} \end{cases} \tag{2}$$

P_i denotes the close price at day "i"

It is worth mentioning, as we can see in equation 2, that a neutral signal do not provide any return. That issue represents a disadvantage with respect to the practitioners or other trading systems. This is because usually the return of a neutral signal is a risk-free return given by the country target of the analysis (e.g the US Treasury Bills). It is also possible to use a fitness function based on return and risk as the Sharpe ratio, however we considered that technical indicators are

supposed to control risk in some way (e.g. break of support levels). Furthermore we want to collate the GA and GE TSs, so it is more suitable in order to evaluate the differences. Nevertheless, we will test different fitness functions in a future work.

3.2 Trading System

The GE trading system is managed by six TI and the six weights corresponding to each indicator (W_1, W_2, W_3, W_4, W_5 and W_6) . In this work the selected TIs are Moving Average crossover (MA), Volume-Price (VOL), Moving Average Convergence Divergence (MACD), Support and Resistances (SR) and Relative Strength Index used in two ways, to spot divergences that show if a trend is fading (RSId), and to identify Overbought/Oversold levels (RSIo). Each one of these technical indicators gives us a signal of buy, sell or neutral. The values of the weights represent the importance of each indicator in obtaining the investment signals of the TA. We have chosen the above indicators guided by the their utility in the world of professional finances and literature. If readers are interested in obtaining complete descriptions of the TIs that have been cited here, we refer them to [6] where these indicators are widely explained.

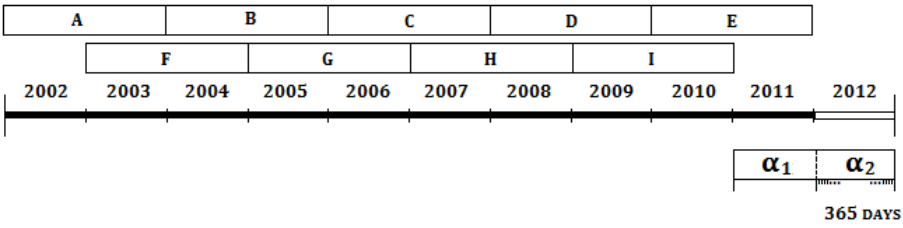


Fig. 1. Data set divisions. The Data set is divided in nine groups (A,B,C,D,E,F,G,H and I) corresponding each to a two years period.

Systems trained with series of historical data are based on the theory that there are continuous and repetitive patterns in the historical price series. Training with this data aims to identify these patterns with the target of building a solution that allows a TS to apply the achieved solutions in the future. These patterns could be short or long series located in any time of the historical data. Based on this theory, and in an attempt to supply a proper environment to extract the best patterns, we proceed in the following way. We divide the data set in nine groups Y_j, (A,B,C,D,E,F,G,H and I) corresponding each to a two years period (see Figure 1). For each company we apply our GE algorithm and we extract a set of rules for each two pairs of years. When we have the collection of rule sets, we test all of them for the last year (α_1). We apply the set of rules with higher performance in the data series to obtain the signal prediction for the

next day. We repeat this process for each day we want to forecast using all the data already available until the day to predict (α_1 plus the daily prices already forecasted of α_2). This methodology can be formalised in the following way:

1. The investor selects a set of Technical Indicators for TA
2. Establish $Weights$ ranges
3. Define a Grammar in BNF
4. For each company i
   (a) For each 2 years Data Y_j
      i. Obtain a GE solution $S_i[Y_j]$
      ii. Obtain a set of rules $Rule_i[Y_j]$ by decoding $S_i[Y_j]$
   (b) For each day D_i in 2012 and For each $Rule_i[Y_j]$
      i. Compute $Returns_{ij}$ of $Rule_i[Y_j]$ on Data $[2011 + D_i - 1]$
5. FinalRule = $Rule_i[Y_j]$ with maximum $Returns_{ij}$

3.3 GE Implementation

We use an integer codification and allow our individuals to generate the offspring by the crossover operation, crucial for the right working of the algorithm as ONeill demonstrates in [18]. We use the single point crossover which has already been demonstrated to be a successful crossover operator and capable to achieve good performance in terms of utility in the process of interchanging blocks in the chromosomes. Finally, we use also the wrapping operator. Wrapping is an advantageous operator as Huggoson shows in the results of [13]. As we explained before, GE uses grammars to build a set of rules that guide our TS. We tested several grammars, and for simplicity we expose the following grammar as the simplest example we are testing:

```
<code>      ::=  <code><indicator> | <indicator>
<indicator> ::=  <MA> | <MACD> | <RSId> | <RSIo>| <SR> | <VOL>

<MA>   ::=   "MA("<short>","<long>","<weigh>");"
<MACD> ::=   "MACD("<short>","<long>","<signal>","<weigh>");"
<VOL>  ::=  "VP("<digit>","<weigh>");"
<SR>   ::=   "SR("<long>","<long>","<weigh>");"
<RSIo> ::=   "RSI(false,"<short>","<long>","<short>","<weigh>");"
<RSId> ::=   "RSI(true," <short> ","<short>","<weigh> ");"

<weigh> ::=  <GECodonValue(1,4)>
<short> ::=  <GECodonValue(1,49)>
<long>  ::=  <GECodonValue(50,99)>
<signal>::=  <GECodonValue(60,89)>
<digit> ::=  <GECodonValue(0,9)>
```

4 Experimental Results

4.1 Data Set

Being the objectives of the paper not only to test [7] with new data, but also to compare it with a new GE approach.As we want a fair comparation between both systems, the set of indicators was adjusted. Thus, we use six technical indicators for both systems (explained in the last section).

Data2012. The trading system is used to invest in the market through the analysis of historical data. The selection of historical data is an important decision in the development of the system, the quality of the results are biased by stability and how profitable are the trends of the market in the analised period. Thus, it should be easier to achieve better returns in a period of rising market than in a diminishing market. So, the already difficult and exciting challenge of building a trading system able to predict the intricate behavior of the stock market becomes more difficult and interesting in a hostile environment. Thus, we test our trading system with current data and analise its performance in one of the most hostile scenarios, the current recession economy. We updated our old dataset that comprised the daily data of the years ranging from 1994 to 2004. Thus, our current data is located between the 2001 and 2012, in the middle of one of the most important economy cracks. We are fixing the period, which answers the question *When*. Nevertheless there is another important parameter to choose the historical data, that is *where*. Our old data is retrieved from the SP&500 which is one of the most important indexes of the stock market. Now, we are considering a group of listed companies in Spain, one of the countries with more economic problems in this economy recession. The companies that form our dataset have been listed with Orbis [12] and have been chosen according to the next criterions:

- Publicly Listed companies of Spain.
- Active companies with at least 5 last years of historical data in the stock market.
- Current market capitalisation greater than 25.000.000 .
- Without pension fund.
- Without financial firms (Financial and insurance activities).
- Without Public administration.
- Without Activities of extraterritorial organism.

Due to lack of time, our experiments are based on a small sample of the data explained above. We select three random companies to check the results in both scenarios (GA and GE). Later, we carry out another experiment with a broader set of nine random companies. The historical series of data have been downloaded by Bloomberg software. We use a daily frequency data of common shares. We refer to this Data set as *Data2012* when testing the [7] GA and our GE proposal.

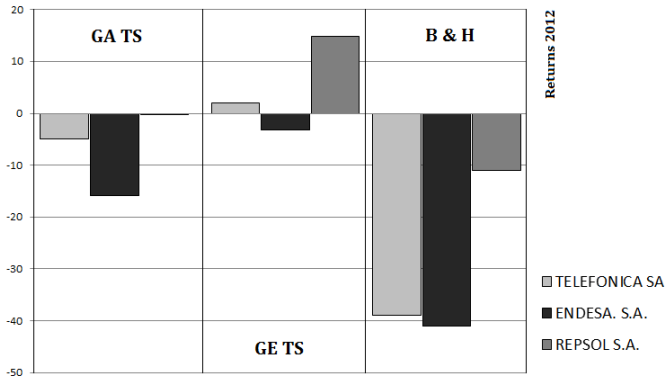


Fig. 2. Returns of three spanish companies with the GA TS, the GE TS and the B&H strategy

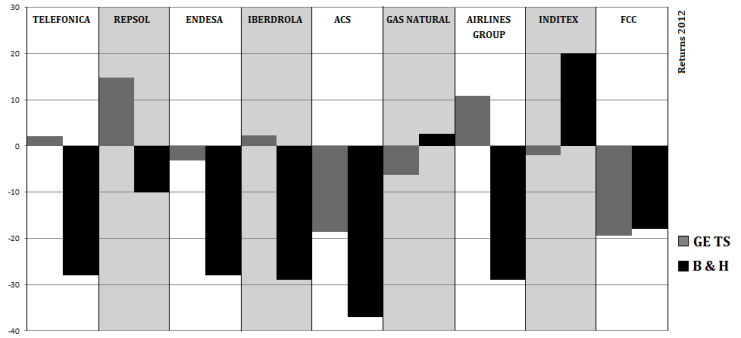


Fig. 3. Returns of nine spanish companies with the GE system and B&H strategy

4.2 GA vS GE on Data2012

Figure 2 shows the results of three enterprises from the Spanish listed companies. The figure is divided in three main sections. First, the left side presents the GA TS returns on 2012, secondly the middle section provides the returns of the GE TS in the same year, and finally the left side represents the profits of the well know Buy and Hold (B&H) strategy. The TS implemented with GE gets profits around the 14%, while the GA can not get earnings, reaching a loss of about 20%. The B&H strategy gets the worst profits, with losses exceeding 90%. We can also see that the two first sections are proportional, so the lower return is provided in the last enterprise, the better result falls in the second company and the first one is located in a middle term.

Figure 3 shows the analysis of an extended set of enterprises related with the GE TS. On one hand, the gray bars represents the profits of the GE TS. On the other hannd the black bars provide the returns of the B&H strategy. The aim of this experiments is to check the genneral profits in a bigger sample of enterprises.

We carry out the same methodology, explained in the previous sections, with nine selected companies from Spain. The general profit of the investment that GE TS is able to achieve is very low, however if we compare against B&H is still much more profitable. Furthermore we can observe a significant fact in this graph. The companies that produce higher losses are both companies related with the building world. Construction companies are the most affected companies in this recession period [21], so this point encourages us to follow with the present research line. In our next researching step we will develop a TS able to chose the set of companies in which we should invest. If we could select the companies with better situation, the investor returns could be multiplied.

5 Conclusions

In this paper we present a Grammatical Evolution Trading system. We tested its behaviour on real data from 2012 for Spanish Companies. In addition we compare the GE system with the work in [7] where the authors applied a GA approach. The TS implemented with GE gets profits around the 14%, while the GA can not get earnings, reaching a loss of about 20%. The analysis with a extended set of nine selected companies from Spain shows that the general profit of the investment is greater than the B&H strategy. Futhermore, we can observe that the companies that produce higher losses are companies related with construction. Those companies are among the hardest hit in this recession period. Thus, in future works we analyse this issue closer and we will test more intensive experiments in our data set.

Acknowledgements. Iván Contreras is supported by Spanish Government Iyelmo INNPACTO-IPT- 2011-1198-430000 project.The work has also been supported by Spanish Gov- ernment grants TIN 2008-00508 and MEC CONSOLIDER CSD00C-07-20811.

References

1. Adamu, K., Phelps, S.: Modelling financial time series using grammatical evolution. In: Proceedings of the Workshop on Advances in Machine Learning for Computational Finance, London, UK (2009)
2. Adamu, K., Phelps, S.: Coevolution of technical trading rules for high frequency trading. In: Proceedings of the World Congress on Engineering, pp. 96–101 (2010)
3. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51(2), 245–271 (1999)
4. Bodas-Sagi, D., Soltero, F., Hidalgo, J., Fernández, P., Fernandez, F.: A technique for the optimization of the parameters of technical indicators with multi-objective evolutionary algorithms. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2012)
5. Brabazon, A., O'Neill, M.: Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. Computational Management Science 1(3), 311–327 (2004)
6. Colby, R.W., Meyers, T.A.: The encyclopedia of technical market indicators. Irwin, New York (1988)

7. Contreras, I., Hidalgo, J.I., Núñez-Letamendia, L.: A GA Combining Technical and Fundamental Analysis for Trading the Stock Market. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 174–183. Springer, Heidelberg (2012)
8. Contreras, I., Jiang, Y., Hidalgo, J., Núñez-Letamendia, L.: Using a gpu-cpu architecture to speed up a ga-based real-time system for trading the stock market. In: Soft Computing - A Fusion of Foundations, Methodologies and Applications, pp. 1–13 (2011)
9. Contreras, I., Jiang, Y., Hidalgo, J.I., Núñez-Letamendia, L.: Using a gpu-cpu architecture to speed up a ga-based real-time system for trading the stock market. Soft Comput. 16(2), 203–215 (2012)
10. Dempsey, I., O'Neill, M., Brabazon, A.: Grammatical Constant Creation. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 447–458. Springer, Heidelberg (2004)
11. Dempsey, I., O'Neill, M., Brabazon, A.: Adaptive trading with grammatical evolution. In: Proceedings of the 2006 IEEE Congress on Evolutionary Computation, July 6-21, pp. 9137–9142. IEEE Press, Vancouver (2006)
12. `https://orbis-bvdinfocom.ezxy.ie.edu.Orbis`
13. Hugosson, J., Hemberg, E., Brabazon, A., O'Neill, M.: Genotype representations in grammatical evolution. Appl. Soft Comput. 10(1), 36–43 (2010)
14. Jiang, Y., Núñez, L.: Efficient market hypothesis or adaptive market hypothesis? a test with the combination of technical and fundamental analysis. In: Proceedings of the 15th International Conference. Computing in Economics and Finance, University of Technology, Sydney, Australia (July 2009)
15. Lohpetch, D., Corne, D.: Discovering effective technical trading rules with genetic programming: Towards robustly outperforming buy-and-hold. In: NaBIC, pp. 439–444. IEEE (2009)
16. Lohpetch, D., Corne, D.: Multiobjective algorithms for financial trading: Multiobjective out-trades single-objective. In: IEEE Congress on Evolutionary Computation, pp. 192–199. IEEE (2011)
17. Núñez, L.: Trading systems designed by genetic algorithms. Managerial Finance 28, 87–106 (2002)
18. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers (2003)
19. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical Evolution: Evolving Programs for an Arbitrary Language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, p. 83. Springer, Heidelberg (1998)
20. Saks, P., Maringer, D.: Evolutionary Money Management. 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. 162–171. Springer, Heidelberg (2009)
21. Díaz-Giménez, J., Dolado, J.J., Bentolila, S., Boldrin, M.: La crisis de la economía española. Análisis económico de la gran recesíon. Monografias fedea (2012)
22. Soltero, F.J., Bodas-Sagi, D.J., Fernández-Blanco, P., Hidalgo, J.I., de Vega, F.F.: Optimization of technical indicators in real time with multiobjective evolutionary algorithms. In: Soule, T., Moore, J.H. (eds.) GECCO (Companion), pp. 1535–1536. ACM (2012)

A Card Game Description Language

Jose M. Font[1], Tobias Mahlmann[2], Daniel Manrique[1], and Julian Togelius[2]

[1] Departamento de Inteligencia Artificial, Universidad Politcnica de Madrid. Campus de Montegancedo, 28660, Boadilla del Monte, Spain
{jfont,dmanrique}@fi.upm.es

[2] Center for Computer Games Research, IT University of Copenhagen, Rued Langaards Vej 7, 2300 Copenhagen, Denmark
{tmah,juto}@itu.dk

Abstract. We present initial research regarding a system capable of generating novel card games. We furthermore propose a method for computationally analysing existing games of the same genre. Ultimately, we present a formalisation of card game rules, and a context-free grammar $G_{cardgame}$ capable of expressing the rules of a large variety of card games. Example derivations are given for the poker variant *Texas hold 'em*, Blackjack and UNO. Stochastic simulations are used both to verify the implementation of these well-known games, and to evaluate the results of new game rules derived from the grammar. In future work, this grammar will be used to evolve completely novel card games using a grammar-guided genetic program.

Keywords: Game design, game description language, evolutionary computation, grammar guided genetic programming, automated game design.

1 Introduction

The ruleset is essential to any game, defining its mechanics and in many ways being the "core" of the game. Rules can't be removed or edited without changing a game's computational properties. The digital and unambiguous nature of most rules (rules for computer games) makes them easy to model as program code. The modelling of rules can serve several different purposes, but the two most prominent are computational analysis (of gameplay) and generation of new games. Computational analysis uses existing games to simulate many playouts under various circumstances to analyse the balance, depth, variety and other aspects of a game. One approach to game (rules) generation may be done by searching a space of games, expressed in some *game description language* (GDL) to find games with desirable properties. These two purposes go well together, as modelling several related games is a good way of constructing a GDL, and computational analysis is typically used to evaluate candidate games when generating novel game rules.

Card games seem to be an interesting application for automated design and computational analysis for several reasons. An important factor is clearly their

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 254–263, 2013.
© Springer-Verlag Berlin Heidelberg 2013

ubiquity and popularity; different card games originated from many parts of the world, and have been played since hundreds of years. Another important factor is their computational simplicity: most card games could be simulated with very limited computational effort compared to games which are designed to be played with a computer and are normally very calculation heavy (e.g. simulation games). Card games share these two aspects with another type of games: classic board games such as Chess, Go and Backgammon. But unlike board games, card games share a common prop: the classic French deck of cards (52 cards of four colours and two jokers). Most card games (certainly most Western card games) can be played using only one or two such decks, and perhaps a few tokens representing money or score. This enables us to model a large variety of card games by simply altering the form of their rules.

In recent years, several authors have attempted to formally describe and automatically generate game rules. The two main game genres where this has been attempted are board games and simple 2D arcade games. In board games, the early work of Hom and Marks [7] was followed by Browne's *Ludi* system, which managed to evolve a board game of sufficient novelty and quality to be sold commercially [2,1]. A system generating simple arcade games was proposed by Togelius and Schmidhuber [11], which was followed up by work by Smith and Mateas [10] and Cook and Colton [3]. In a related genre, Mahlmann et al. have defined a GDL for turn-based strategy games [9]. Mahlmann et al. also published similar work to our approach, evaluating different game mechanics of the card game *Dominion* [8]. The representation of game rules and level of abstraction in these examples varies wildly, from expression trees reminiscent of those used in genetic programming, to tables of actions and consequences, to first-order logic expressed in AnsProlog. In all of these examples, the space of games was searched using evolutionary algorithms, except Smith and Mates who use answer set programming and constraint solving. An overview of this line of research can be found in a recent tutorial by Nelson[1].

There are several important considerations when devising a GDL for a particular game genre. Prominent among these is the expressivity of the language: all games in scope have to be expressible in the language. Normally opposing the expressivity stands the compactness or verbosity of the language, i.e. its human-readability, and how efficiently the search space can be traversed by using for example an evolutionary algorithm. Especially the latter is only marginally explored, and more research needs to be done.

2 Definition of a Search Space for Card Games

It is not possible to evolve any kind of game without setting the constraints that define the basics of the search space [11]. Insofar as card games are the subject of the evolutionary process described here, it is mandatory to define a set of axioms that will be shared by any card game in the evolutionary population.

[1] `http://kmjn.org/`

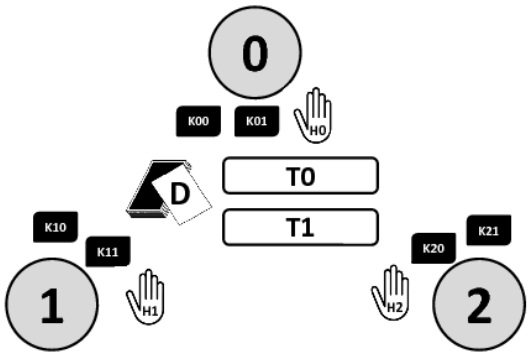


Fig. 1. Main components of a card game with three players and two table locations ($P = 3$ and $T = 2$)

Those axioms are the following:

- The card game is played by exactly P players.
- A *card location* is defined as a place where any number of cards can be placed during the game. Every game has the following card locations L:
  - A number of hands H, one for each player in P. Hereafter, HX refers to the "hand of the current player" and HA to "hands of all other players". $H0, H1, H2$ refer to the hand of player one, two, or three respectively.
  - One standard French deck of cards D, composed of four suits with 13 cards each, numbered from 1 to 13 (jokers are excluded). Cards in the deck are always placed face down.
  - A number of table locations T, which are areas in a virtual table where cards can be placed, always face up.
- Insofar as many card games involve bets, *tokens* are defined as virtual representations of coins or chips. Analogously to card locations, a token location K is defined as a place where any number of tokens can be placed during the game. Every player J has two token locations, $KJ0$ and $KJ1$. $KJ0$ functions as a player's stash), and $KJ1$ may be used to place bets. To illustrate the nomenclature, player 0 has two token locations denoted by $K00$ and $K01$. The term $KX1$ refers to "the current bet of the current player", and $KA1$ to "the current cumulative bets of all other players". Figure 1 shows the main components of a card game with $P = 3$ and $T = 2$, therefore including six token locations ($K00, K01, K10, K11, K20, K21$), and three hands ($H0, H1, H2$).
- A card game consists of several stages, each one composed by a set of rules. The stages are played in sequential order.
- Stages are played out in a turn-based way with a round-robin turn order. Within each stage and during his turn, a player may play a variable number of rules. A player's turn is over when he is *done*, *next* or *out*:
  - *done* refers to a player being done with the current stage (but will return to play in the nest stage).

  - *next* indicates that the player has finished his turn but will be able to play again during the current stage when all other players' turns are over.
  - *out* defines a player as being out of the game. He will not play in any remaining stages of the game.
- A stage ends when all players are either *done* or *out*. While one or more player are *next*, their turns alternate in a round-robin order.
- A game ends when all stages have been played or when all players are *out*.

Given these specifications, a card game is defined as a set of stages, a ranking of card combinations, and a set of winning conditions. Every stage comprises a set of conditional rules (production rules) in the form "if antecedent then consequent". Antecedents are conditions that - when fulfilled - trigger actions defined in consequents. For example: "if the player has no cards, then he draws a card". Optionally, a rule may have no antecedents at all, meaning that it can be played without any particular prerequisites. Following this structure, three different kinds of rules can be found:

- *Optional rule.* Standard rule which can be played a multiple number of times. If the current player satisfies the condition defined in the antecedent, he can play the action defined in the consequent.
- *Play-only-once rule.* These rules (marked with a "once" modifier) can only be played once per stage.

Table 1. Types of antecedents and their related conditions

Antecedent	Condition	Example
tokens, KA, RESTRICTION, KB	Compares the amount of tokens in token location KA with the amount of tokens in KB. Condition is fulfilled if the comparison satisfies the RESTRICTION ($<, >, =, \leq, \geq$)	tokens, $K01$, $\geq$, $K11$
play, LA, RESTRICTION, LB	Compares the play in card locations LA with the play in LB. Condition is fulfilled if the comparison satisfies the RESTRICTION ($<, >, =, \leq, \geq$)	play, $H0 + T0$, $<$, $H1 + T0$
sum, LA, RESTRICTION, LB	Sums up the numbers of the cards in locations LA and the ones in LB. Condition is fulfilled if the comparison satisfies the RESTRICTION ($<, >, =, \leq, \geq$)	sum, $H0$,=,$H1$
have, COMBINATION	Check if the player has a given COMBINATION of cards, suits or numbers in his hand.	have, 2 of diamonds
draw	Draw one card from the deck and place it in the hand of the current player	draw
show, RESTRICTION, LA	Show a set of cards from player's hand that satisfy a given RESTRICTION ($<, >, =, \leq, \geq, same\ suit, same\ number$) when compared with a card in LA.	show, card with same suit, $T0$
λ (LAMBDA)	No condition to satisfy.	unconditional

– *Mandatory rules.* Mandatory rules must be played (and only once) by every player at the beginning of his first turn of the stage. They are marked with a "mandatory" modifier.

Additionally, so called *computer commands* are always played once by the game (i.e. a virtual dealer) at the beginning of the stage before the players' turns. These rules are marked with a "com" modifier. Command rules include no conditions, i.e. no antecedent, and only one consequent which describes the action to be played. Table 2(b) illustrates the implemented types of computer rules.

Once the game's setup is finished, the first player's turn begins. After all "mandatory" rules have been invoked (which may be none), he may play as many "once" and/or optional rules as possible. When there are no satisfiable antecedents left, the player is marked as *done*. Optionally, a player may choose to not play any (more) rules, deliberately changing his state to *next* (if he wants to play again during the current stage) or *done* (otherwise). Table 1 shows the different kinds of antecedents possibly involved in a rule. Notice that the terminal symbol "lambda" (λ) is used to represent a "null" value for any given variable, and LA and LB refer to any two valid card locations.

When played, rules trigger consequents which modify the game state. Table 2(a) shows the types of consequents which can be found in a rule. Notice that the "bet" consequent does not specify any location. Betting always refers to moving tokens from a player's total amount of tokens to his current bet. "Gain" always implies moving tokens from the specified location to a player's total amount of tokens.

Every card game has its own card and play values. Those features are specified in the ranking: a table that contains pairs in the form [*play*, *score*], in order to assign a fixed *score* to a given *play* (card combination). This table is used every time a condition "play" is evaluated in order to know the actual value of the card combinations in comparison. For example, when playing poker, all valid poker hands have to be indexed in the ranking table, therefore it is possible to know that two pairs have a lower value than three of a kind. When a given hand has no specified value, the standard ranking of cards is assumed. Thus: $1 > 13 > 12 > 11 > 10 > 9 > 8 > 7 > 6 > 5 > 4 > 3 > 2$. Upon finishing the game, winning conditions determine which player wins. This is done by assigning points to the remaining amount of tokens the player has and extra points if the player status is not *out*. An exception here is, that certain consequents may declare a certain player as the winner a priori.

2.1 The Card Game Language

A context-free grammar (CFG) G is defined as a string-rewriting system comprising a 4-tuple $G = (S_N, S_T, S, P)/S_N \bigcap S_T = \emptyset$, where S_N is the alphabet of non-terminal symbols, S_T is the alphabet of terminal symbols, S represents the start symbol or axiom of the grammar, and P is the set of production rules in Backus-Naur form. Every grammar generates a language composed where all sentences are grammatically valid, meaning they conform to the constraints defined by that grammar. The context-free grammar $G_{cardgame}$ has been designed

Table 2. Overview of rule Consequences

(a) Types of consequents in a rule

Consequent	Action	Example
pifr, LA * AMOUNT, FACE	Draw a given AMOUNT of cards from location LA. FACE indicates if cards are drawn face up or down.	pifr, $D*2$, down
puin, LA * AMOUNT, RESTRICTION, FACE	Put a given AMOUNT of cards from player's hand in card location LA, only if those cards satisfy a given RESTRICTION ($<, >, =, \leq, \geq$) .	puin, $T0*1$, $>$, up
bet, RESTRICTION, KX	Bet an amount of tokens that satisfy a given RESTRICTION ($<, >, =, \leq, \geq$, *lambda*) when compared with the amount of tokens in location KX.	bet, $>$,$K01$
gain, KX	Gain the amount of tokens in token location KX.	gain, $K21$
playit	Place the set of cards specified in the antecedent in the location specified in the antecedent	-
next	The player status is set to *next*.	-
done	The player status is set to *done*.	-
out	The player status is set to *out*.	-
win	The player immediately wins and the game ends.	-
end	The game immediately ends.	-

(b) Special Consequences in computer rules

Rule	Action	Example
com_deal, PLAYERS, AMOUNT	deal a given AMOUNT of cards from the deck to a set of PLAYERS	deal, player 1, 4
com_deal, $L0$, AMOUNT	deal a given AMOUNT of cards from the deck to location $L0$	deal, $T0$, 4
com_give, PLAYERS, AMOUNT	give a fixed AMOUNT of tokens to a set of PLAYERS	give, players 0 and 1, 100

to generate the language composed by all the valid card games that comply with the axioms described above.

A grammar-guided genetic program (GGGP) is an evolutionary system that could potentially find solutions to any problem whose syntactic restrictions can be formally defined by a CFG [5]. Individuals are derivation trees of the CFG that, when the algorithm starts, are generated by a grammar-based initialization method [6]. Neither this method nor crossover and mutation operators can generate invalid individuals because they are not contained in the language described by the CFG [4]. Thus the individual population of a GGGP using $G_{cardgame}$ is a set of derivation trees, each of them defining a card game that follows the above

Table 3. Codification of Texas hold'em poker stages

STAGES

NAME	RULES	NAME	RULES
STAGE 0	com _ deal, ALL PLAYERS, 2	STAGE 6	com _ deal, T1, 1
(pre-flop)	com _ give, ALL PLAYERS, 99	(turn)	com _ deal, T0, 1
STAGE 1	mandatory_unconditional_bet, λ, λ	STAGE 7	unconditional_bet, $\geq$, $KA1$
(first bet)	unconditional_bet, $=$, $KA1$	(3rd bet)	once_unconditional _ *next*
	once_unconditional _ *next*		once_unconditional_*done*
	once_unconditional_*done*	STAGE 8	mandatory_tokens, $KX1$, $<$, $KA1$
STAGE 2	mandatory_tokens, $KX1$, $<$, $KA1$	(check bets)	_ *out*
(check bets)	_ *out*	STAGE 9	com _ deal, T1, 1
STAGE 3	com _ deal, T1, 1	(river)	com _ deal, T0, 1
(flop)	com _ deal, T0, 3	STAGE 10	unconditional_bet, $\geq$, $KA1$
STAGE 4	unconditional_bet, $\geq$, $KA1$	(final bet)	once_unconditional _ *next*
(2nd bet)	once_unconditional _ *next*		once_unconditional_*done*
	once_unconditional_*done*	STAGE 11	mandatory_tokens, $KX1$, $<$, $KA1$
STAGE 5	mandatory_tokens, $KX1$, $<$, K _	(check bets)	_ *out*
(check bets)	*out*	STAGE 12	mandatory_play, $HX + T0$, $>$,
		(showdown)	$HA + T0$ _ gain, $KA1$

constraints. Our card games are codified in the genotype of individuals with the structure *STAGES* : *RANKING* : *WINNING CONDITIONS*, where:

- *STAGES* represents several sets of rules, each of them corresponding to a stage of the game.
- *RANKING* is a list of pairs in the form [*play*, *score*] which will be later translated into a ranking table.
- *WINNING CONDITIONS* specifies two natural numbers that are the amount of points awarded to a player for each remaining token and for not being out of the game respectively.

$G_{cardgame}$ has been intentionally designed to generate a high level language containing a great variety of card games. Since individuals of a GGGP have no fixed size, the evolutionary system becomes a flexible tool, being able to design games from simple one-stage games up to long strongly ruled games. Nevertheless, despite the high level approach, this language contains at least three very well known card games: Texas hold 'em poker, Blackjack and UNO. For brevity, the following paragraphs assume that the reader is familiar with said games and their terms.

Table 3 presents Texas hold 'em poker's stages codified as a sentence of the language generated by $G_{cardgame}$. The game is composed by 12 stages which cover the standard parts of the game (pre-flop, flop, turn, river and showdown) as well as additional stages for player bets and bet checking. Bets are checked in order to set the status of players which have folded as *out*. The ranking includes poker hands, and the winning conditions define that the winner is the player with the most tokens at the end of the game. Please note that single cards are not

Table 4. Codification of the two example games Blackjack and UNO

(a) Blackjack

STAGES	
NAME	RULES
STAGE 0	com _ deal, ALL PLAYERS, 2
	com _ give, ALL PLAYERS, 99
STAGE 1	mandatory_unconditional_bet, λ, λ
STAGE 2	unconditional_pifr, D*1, up
	unconditional_*done*
STAGE 3	mandatory_sum, $HX > 21$ _ *out*
	mandatory_sum, $HX > HA$ _ gain, KA

(b) UNO

STAGES	
NAME	RULES
STAGE 0	com _ deal, ALL PLAYERS, 7
	com _ deal, T0, 1
STAGE 1	show, same suit, $T0$ _ playit
	show, same number, $T0$ _ playit
	draw _ *next*
	mandatory_have, λ _ *win*

listed in the ranking table as poker uses the default card ranking. The ranking used for the poker hands is: Straight flush (900), Poker (800), Full house (700), Flush (600), Straight (500), Three of a kind (400), Two pairs (300), and One pair (200).

Table 4(a) shows the codification of blackjack. In this implementation of blackjack, players play against each other instead of the dealer. The winner of the game is the player who earned more tokens during the game, meaning the player who bet most and got the highest score below or equal to 21. The ranking sets all figures' values (13, 12, 11) to 10, and includes the two possible values for an ace: 1 and 10. The only winning condition is one point per token earned.

Table 4(b) shows the codification the basic rules of UNO. For simplicity, we excluded cards with any special effect, e.g. changing the turn order or skipping players. Notice that the ranking table is empty because the goal for a player is to empty his hand (rather than getting the best play). There is no need to set winning conditions or rankings because the winner is determined by the last rule of stage 1.

3 Experimental Results

In order to test the quality of the card games generated by $G_{cardgame}$, random plays have been run over a set of games expressed by its language. All games have been designed for three players and up to two table locations ($P = 3, T = 2$). The term "random" herein refers to artificial players playing random moves.

The set of games is composed by the sample games presented above: Texas hold 'em poker, Blackjack and UNO. Each of these has been run a 1000 times. In addition to this, 1000 random mutations of each of them have also been tested. Ultimately, we also sampled 1000 randomly generated card games.

For each game the following data has been collected: number of plays won by each player, number of plays ended in a draw, number of games which crashed and the average number of turns needed to properly finish the game. A game

is considered to be crashed when it hasn't finished within 100 turns or when a non semantic expression occurs, e.g. forcing a player to bet an amount of tokens lower than zero.

Table 5. Results obtained from running Texas hold 'em poker, Blackjack, UNO, random mutations of them and randomly generated games, 1000 times each

Game	Times run	Times won			Times draw	Times crashed	AVG turns to finish
		0	1	2			
Texas	1000	235	324	437	4	0	39
Blackjack	1000	401	282	317	0	0	20
UNO	1000	319	285	294	102	0	59
Random Texas mutations	1000	237	15	15	523	210	13
Random blackjack mutations	1000	49	30	103	516	302	32
Random UNO mutations	1000	79	45	22	293	561	26
Random games	1000	174	32	36	485	273	19

Table 5 shows the results obtained from these tests. The three sample games have been properly played during all 1000 runs, showing that the language defined by $G_{cardgame}$ allows to codify well-formed and semantically valid card games. These games are finished in a very reasonable number of turns, whereas UNO seems to be the longest one. It seems easy to conclude, that this may because UNO's last stage is not finished until one of the players gets rid of all his cards. UNO is also the game most likely to end in draw. A draw takes place when the deck and the table location run out of cards before any player has been able to win.

As expected, random mutations on these games lead to an increase in their crash rate. Nevertheless, almost 80% of poker mutations, 70% of blackjack mutations and 45% of UNO mutations can be properly played until the end of the game. Draw rates also increased, meaning that changes produced by mutations create games where winners are not clearly defined.

Randomly generated games produce similar results, showing a rate of playable games close to 73%.

4 Conclusions and Future Work

A context-free grammar, $G_{cardgame}$, that generates a game description language for card games has been presented. All card games contained in this language share a fixed set of basic axioms, that represent the main components of a game. Every game defines its very own rules that, when played along with these axioms, compose a playable card game. This language contains a set of high-level instructions that allows the codification of a high variety of games. To

show this, examples codifications of Texas hold 'em poker, Blackjack and UNO have been proposed. The experiments conducted show the ability of $G_{cardgame}$ to produce randomly generated but playable card games. Our results indicate, that it is possible to improve the generation of card games with searching and optimizing techniques. For this reason, our grammar is intended to be part of a grammar-guided genetic program that evolves populations of card games in order to automatically generate entertaining and playable card games. Leading the evolutionary process to improve the satisfaction achieved by players when playing these games will allow the possibility of creating interesting card games without human assistance. In other words, mimicking human creativity inside a computer process.

References

1. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games 2(1), 1–16 (2010)
2. Browne, C.: Automatic generation and evaluation of recombination games. Ph.D. thesis, Queensland University of Technology (2008)
3. Cook, M., Colton, S.: Multi-faceted evolution of simple arcade games. In: Proceedings of the IEEE Conference on Computational Intelligence and Games, CIG (2011)
4. Font, J.M., Manrique, D., Ríos, J.: Evolutionary construction and adaptation of intelligent systems. Expert Systems with Applications 37, 7711–7720 (2010)
5. Font, J.M.: Evolving Third-Person Shooter Enemies to Optimize Player Satisfaction in Real-Time. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 204–213. Springer, Heidelberg (2012)
6. Garcia-Arnau, M., Manrique, D., Rios, J., Rodriguez-Paton, A.: Initialization method for grammar-guided genetic programming. Knowledge-Based Systems 20(2), 127–133 (2007)
7. Hom, V., Marks, J.: Automatic design of balanced board games. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), pp. 25–30 (2007)
8. Mahlmann, T., Togelius, J., Yannakakis, G.: Evolving card sets towards balancing dominion. In: IEEE World Congress on Computational Intelligence, WCCI (2012)
9. Mahlmann, T., Togelius, J., Yannakakis, G.: Modelling and evaluation of complex scenarios with the strategy game description language. In: Proceedings of the Conference on Computational Intelligence and Games (CIG) 2011, Seoul, KR (2011)
10. Smith, A.M., Mateas, M.: Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In: Proceedings of the IEEE Conference on Computational Intelligence and Games, Copenhagen, Denmark, August 18–21, pp. 273–280 (2010)
11. Togelius, J., Schmidhuber, J.: An experiment in automatic game design. In: IEEE Symposium on Computational Intelligence and Games, CIG 2008, pp. 111–118. IEEE (2008)

Generating Map Sketches for Strategy Games

Antonios Liapis[1], Georgios N. Yannakakis[1,2], and Julian Togelius[1]

[1] Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
[2] Department of Digital Games, University of Malta, Msida, Malta

Abstract. How can a human and an algorithm productively collaborate on generating game content? In this paper, we try to answer this question in the context of generating balanced and interesting low-resolution sketches for game levels. We introduce six important criteria for successful strategy game maps, and present map sketches optimized for one or more of these criteria via a constrained evolutionary algorithm. The sketch-based map representation and the computationally lightweight evaluation methods are geared towards the integration of the evolutionary algorithm within a mixed-initiative tool, allowing for the co-creation of game content by a human and an artificial designer.

1 Introduction

The games industry has often used procedurally generated content to increase the unexpectedness of a game, and thus its replayability value. As games increase in scale, it is becoming common practice for game designers to use procedural content generation tools during development time in order to limit costs and time requirements. In order to assist such design work, computer-assisted design tools should be able to automate the mechanizable parts of content creation (such as ensuring playability and evaluating game balance) and to optimize, on their own, specific gameplay features deemed significant by the designer. Not only would such a tool increase a designer's creative output, it should be able to incite human creativity through the dialogue between the artificial and the human designer.

This paper presents steps towards actualizing such a tool, using maps for strategy games as its test domain. Six important measures of quality for strategy game maps are introduced, inspired by game design patterns [1] popular within strategy games and by previous experiments on map evolution [12]. These criteria are optimized via evolutionary algorithms and their impact on map generation is evaluated. The map generation component is integrated within a mixed-initiative design tool which allows for the co-creation of strategy maps with designers. With the proposed tool we try to overcome some of the limitations of mixed-initiative design as we rely on simple map sketches to counter user fatigue and designer biases reported in the literature [8].

2 Related Work

While procedural content generation has been used in some games since the eighties, recent trends have seen an increasing use of PCG tools such as *SpeedTree*[1] during

[1] `http://www.speedtree.com`

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 264–273, 2013.
© Springer-Verlag Berlin Heidelberg 2013

development time, to partly automate design work. PCG tools have to balance between expressivity and controllability, with methods capable of producing a wide range of content usually being very hard for designers to work with. Academic interest on more controllable PCG methods is only a few years old [14], but search-based processes [13] such as genetic algorithms are becoming a popular solution to this problem [2,11,4,6].

Maps have often been generated algorithmically in games such as *Civilization* (MicroProse, 1991) and *SimCity* (Maxis, 1989); such games usually rely on tightly designed processes to construct playable maps. Within academia, strategy game maps have also been optimized via stochastic search [3,12] or answer-set solvers [10]; while designers can decide on the objectives or constraints of the generated content prior to the generative process, these projects are hardly interactive. In the authors' previous work, a mixed-initiative tool attempted to address the issues of authorial control and capturing human preferences [8]. The tool failed to achieve its stated goals, primarily due to the requirement of hand-crafting a large-scale initial map, which introduced designer bias and fatigue. In order to counter such limitations, this paper reduces the resolution of the human-authored sketches and allows more user control over the optimized features.

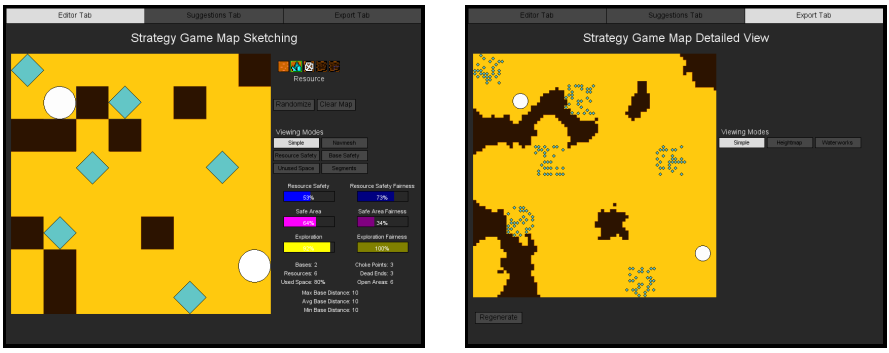
3 Methodology

The tool presented in this paper allows a human or an artificial designer to create low-resolution map sketches. These sketches contain the necessary elements for a simple strategy game, and can be optimized via a genetic algorithm on a number of selected fitness dimensions pertaining to balanced strategic gameplay.

The maps used in this experiment are abstractions of levels used in successful strategy games such as *Starcraft* (Blizzard, 1998). A map is presented to the user as a sketch consisting of a small number of *tiles* (see Fig. 1a). Tiles can be *passable* (light) or *impassable* (dark), and passable tiles can contain *player bases* (circles) or *resources* (rhombi). The map layout assumes that each player starts at a base and gathers resources to produce units; units move through passable tiles to attack the opponent's base.

3.1 Map Design Tool

A graphical interface has been developed to allow a user to manually edit a map sketch (see Fig. 1a). While the sketch is being edited, its scores in several fitness dimensions (see Section 3.2 below) are updated and displayed to the user; the user can also select one or more fitness labels and generate an optimal map on the selected fitness dimensions. At any point, the designer can switch to the final map view (see Fig. 1b), displaying the complete map on which the strategy game can be played. Currently the final map is constructed via random processes and cellular automata to create an organic-looking map which however retains all the properties (chokepoints, passable paths) of the low-resolution sketch. Manual editing, evaluation and evolution (see Section 3.2) are all done on the sketch level, making computations such as pathfinding easier and reducing the required human effort.



(a) Sketching interface while the user generates a rough level sketch

(b) The sketch in Fig. 1a rendered as a complete map, generated with cellular automata

Fig. 1. The User Interface for the mixed-initiative level generation tool

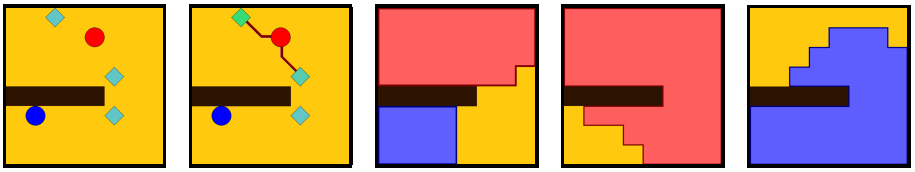
3.2 Evolutionary Optimization

Each map is encoded as an array of integers: each integer represents a tile's type (passable, impassable, base or resource). These parameters are adjusted via constrained optimization — ensuring the playability of feasible maps — carried out by a feasible-infeasible two-population genetic algorithm [5] (FI-2pop GA). FI-2pop GA evolves two populations, one with feasible maps and the other with infeasible maps. Each population selects parents among its own members, but feasible offspring of infeasible parents are moved to the feasible population and vice versa. This interbreeding increases the occurrence of feasible individuals and boosts the feasible population's diversity.

Both populations evolve via fitness-proportionate roulette-wheel selection of parents; parents are recombined using two-point crossover. Mutation can occur on an offspring of two parents (1% chance), or on a single parent (5% chance) in order to create a single-parent offspring. During mutation, 2 to 6 tiles may be transformed: a tile may be swapped with its adjacent (15% chance), an impassable tile can be transformed into passable and vice versa (5% chance) or a passable tile can be transformed into a resource (1% chance). The small number of tiles transformed with each mutation increases the locality (in terms of map structure) of the stochastic search; preliminary tests have shown that when this mutation strategy is combined with the relatively high chance of mutation chosen, the population does not suffer from premature convergence.

The fitnesses used to evolve the feasible population are presented below, while the infeasible fitness is presented at the end of this section.

Feasible Fitnesses. A feasible map sketch is evaluated on six fitness dimensions (see (3) to (8) below) which are inspired by game design patterns [1] suitable for strategic gameplay and geared towards game pace and player balance in terms of starting conditions. Game pace is affected by the *area control* afforded by a player's starting location — including control of *strategic resources* — and by the challenge for enemies to discover this location via *exploration*. If a player has an easily controllable, resource-rich area around their base and their base is difficult to reach by enemies, then defensive



(a) (b) (c) (d) (e)

Fig. 2. Visualization of resource safety, safe areas and exploration on a test map (Fig. 2a) with impassable tiles (dark), resources (rhombi), base 1 at the map's top and base 2 at the bottom. Resources at the map's top are much closer to base 1 than to base 2, and therefore have large $s_{t,1}$ values (connected in Fig. 2b). The bottom-most resource has an equal distance between the two bases, therefore it is contested ($s_{t,1} \approx s_{t,2} \approx 0$). Figure 2c shows the areas around the two bases with safety values over C_s (A_1 at the top, A_2 at the bottom). By applying flood fill from base 1 until base 2 is covered (Fig. 2d) and from base 2 until base 1 is covered (Fig. 2e), we calculate $E_{1\to2}$ and $E_{2\to1}$ respectively.

play is favored and game pace is slow. If a base is within the enemy's reach, and resources are in contested and difficult to control areas, then fast-paced aggressive play is favored. On the other hand, *player balance* is a universal design pattern for any multiplayer game [1], and is captured in this paper as the *symmetry* in affordances for game pacing among players. The concepts of safety, fairness and path overlap presented in this paper have been covered in previous work [12], but they are evaluated differently: while the safety metric in (1) is similar, base safety and exploration are only loosely captured in [12] by *base space* and *base distance*, respectively; more emphasis is also placed on balance, with three fitnesses rather than the single *resource fairness* of [12].

In order to evaluate area control and exploration, two heuristics for *safety* and *exploration* are used in the calculation of the fitnesses in (3)–(8). The safety metric ($s_{t,i}$) in (1) evaluates a tile t according to its safety with respect to a player base i; the closer the tile is to base i compared to any other base, the larger its safety value. The exploration metric (E_i) evaluates the effort required to discover all other player bases from base i; it uses a simple flood fill algorithm to simulate random exploration of the map, which ends once one base is discovered and runs again for every other base. It is calculated as per (2), and has high values for distant bases and if open areas exist between bases.

$$s_{t,i} = \min_{\substack{1\le j\le N_B \\ j\neq i}} \left\{ max\left\{0, \frac{d_{t,j} - d_{t,i}}{d_{t,j} + d_{t,i}}\right\}\right\} \tag{1}$$

$$E_i = \frac{1}{N_B - 1} \sum_{\substack{j=1 \\ j\neq i}}^{N_B} \frac{E_{i\to j}}{w_m h_m - N_I} \tag{2}$$

where N_B is the number of bases; $d_{t,i}$ is the distance from tile t to base i (using A* pathfinding); w_m and h_m is the map's width and height, respectively; N_I is the number of impassable tiles and $E_{i\to j}$ is the map coverage when a four-direction flood fill is applied starting from base i and stopping once base j has been found (see Fig. 2).

The *resource safety* fitness (f_{res}) in (3) uses the safety metric from (1) to calculate the safety of the map's resources. Low scores in this fitness correspond to maps with resources in contested areas, equally accessible to two or more player bases. The *base safety* fitness (f_{saf}) in (4) calculates the safe areas around every player's base. Low scores in this fitness correspond to maps with insecure bases, since many areas around them are easily accessible by at least one enemy base. The *exploration* fitness (f_{exp}) in (5) uses the exploration metric from (2) to simulate the difficulty in finding other bases from each player's base.

$$f_{res} = \frac{1}{N_R} \sum_{j=1}^{N_R} \max_{1 \leq i \leq N_B} \{s_{t_j,i}\} \tag{3}$$

$$f_{saf} = \frac{1}{w_m h_m - N_I} \sum_{i=1}^{N_B} A_i \tag{4}$$

$$f_{exp} = \frac{1}{N_B} \sum_{i=1}^{N_B} E_i \tag{5}$$

$s_{t_j,i}$ is the safety metric of resource j (located at tile t_j) to base i; A_i is the map coverage of safe tiles for base i and E_i is the exploration metric for base i. A tile t is safe for base i if its $s_{t,i} < C_s$; the constant $C_s = 0.35$ throughout this paper, as it amounts to a good ratio of contested areas in most maps (see Fig. 2c).

Fitnesses in (3)–(5) do not differentiate between players; for instance, high scores in f_{res} can correspond to a map where all resources are safe for only one base. Since competitive strategy games favor equivalent starting conditions for each player, the fitnesses in (6)–(8) are evaluated with regards to player balance. Thus, resource balance (b_{res}), base safety balance (b_{saf}), and exploration balance (b_{exp}) are calculated as follows:

$$b_{res} = 1 - \frac{1}{N_R N_B (N_B - 1)} \sum_{k=1}^{N_R} \sum_{i=1}^{N_B} \sum_{\substack{j=1 \\ j \neq i}}^{N_B} |s_{t_k,i} - s_{t_k,j}| \tag{6}$$

$$b_{saf} = 1 - \frac{1}{N_B (N_B - 1)} \sum_{i=1}^{N_B} \sum_{\substack{j=1 \\ j \neq i}}^{N_B} \frac{|A_i - A_j|}{max\{A_i, A_j\}} \tag{7}$$

$$b_{exp} = 1 - \frac{1}{N_B (N_B - 1)} \sum_{i=1}^{N_B} \sum_{\substack{j=1 \\ j \neq i}}^{N_B} \frac{|E_i - E_j|}{max\{E_i, E_j\}} \tag{8}$$

Infeasible Fitness. Infeasible maps fail to satisfy playability constraints (having unreachable bases or resources) or designer specifications regarding the number of map features. In order to increase the chances of infeasible parents creating feasible offspring, the infeasible population must shift its members towards the border with feasibility [9]. The infeasible population optimizes its members according to the infeasible fitness (f_{inf}) in (9), which aims to minimize the distance from feasibility. This distance

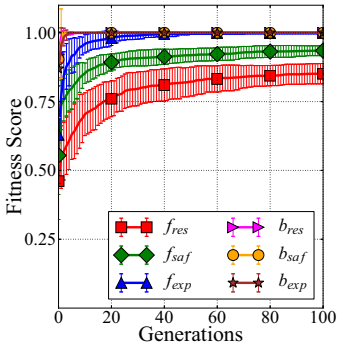


Fig. 3. Optimization of the populations' maximum fitness scores when evolving strategy maps on a single fitness. Error bars represent standard deviation across 20 runs.

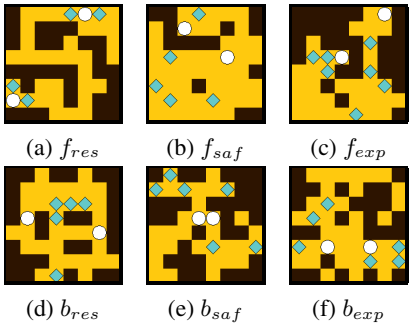


Fig. 4. Best final individuals among 20 evolutionary runs, optimized for a single fitness dimension displayed in each map's caption

from feasibility has four components, equal to the number of constraints for feasible maps: a) fidelity with designer-specified number of bases, b) fidelity with designer-specified number of resources, c) passable paths between bases and d) passable paths between resources and bases.

$$f_{inf} = 1 - \left[\frac{1}{4}|N_B - N_{B,d}| + \frac{1}{4}|N_R - N_{R,d}| + \frac{1}{4}\frac{2d_b}{N_B(N_B-1)} + \frac{1}{4}\frac{d_r}{N_R N_B}\right] \quad (9)$$

where $N_{B,d}$ and $N_{R,d}$ is the designer-specified allowed number of bases and resources, respectively, and d_b and d_r is the number of base pairs and base-resource pairs that are not connected, respectively.

4 Experiments

Several experiments were conducted to test the efficiency of the genetic algorithm used to optimize the rough map sketches of the mixed-initiative tool. These experiments were conducted without any user interaction or human-authored sketches, and assess the algorithm's ability to optimize one or more fitness dimensions. All experiments presented below run for 100 generations, on a population of 100 individuals including both feasible and infeasible genes; the number of individuals is large enough for the simultaneous optimization of two populations while allowing for sufficiently fast evolutionary runs as demanded by a responsive mixed-initiative tool. Evolving maps have 64 tiles (with equal width and height of 8 tiles), 2 bases and anywhere between 4 and 10 resources.

4.1 Optimizing a Single Fitness Dimension

Using a single fitness dimension as the objective function, the genetic algorithm aims to optimize a single gameplay feature; the generated maps are, thus, expected to be

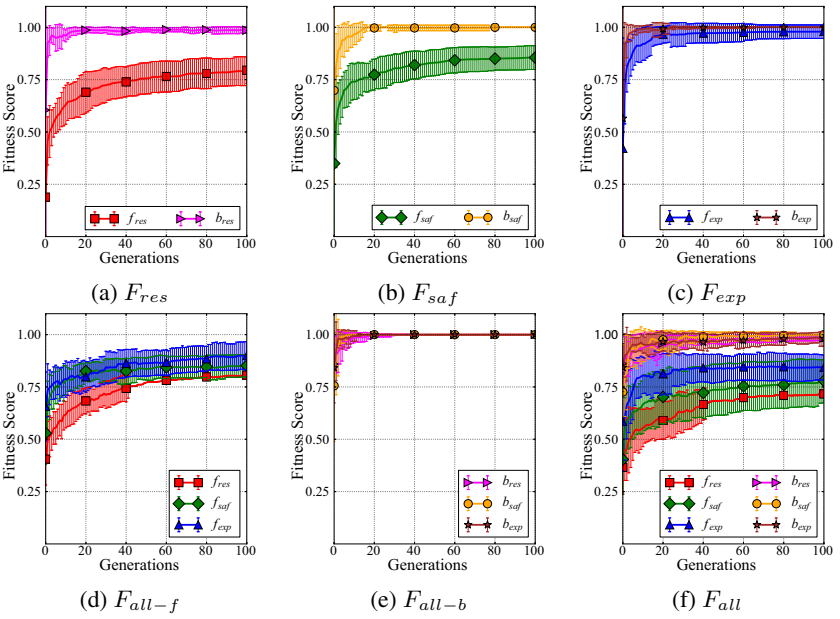
one-sided and lack the necessary features for competitive strategy play. Figure 3 shows the evolutionary progress of the maximum fitness in the population; displayed values are averaged from 20 independent runs, with the standard deviation shown as error bars. The highest scoring final individual among the 20 runs for each fitness dimension is shown in Fig. 4. Results indicate that optimal fitness scores for balance (b_{res}, b_{saf}, b_{exp}) are easily attainable, since high-scoring individuals exist even in the random initial populations and evolution quickly finds optimal solutions for these fitness dimensions within few generations. Observing the maps in Fig. 4, optimal maps in b_{res} have symmetrical resources between players; resources are often far from player bases, so that differences between their distances from each base remain relatively small. Optimal maps in b_{saf} have bases near each other and safe areas are small but equal between the bases. Optimal maps in b_{exp} often have bases near each other (even adjacent), and finding the other base is equally effortless for either player. Among the other dimensions, f_{res} is the most difficult to optimize, mainly because of how the safety metric is calculated: based on (1), $s_{t,i}$ cannot reach its optimal value (1.0), but approaches it if $d_{t,j} \gg d_{t,i}$ for all bases $j \neq i$. Granted that the small size of the maps cannot allow such disparities between distances, it is expected that even in the best circumstances $s_{t,i}$ and thus f_{res} will be considerably lower than 1.0. Similarly, f_{saf} cannot reach optimal values since there will be at least one tile in the map at equal distance from both bases (and thus not safe). The best maps in f_{res} have resources adjacent to bases, while the bases are far from each other. The best maps in f_{saf} have numerous impassable regions between bases; in Fig. 4b, one base is hidden behind impassable tiles while the other base has safe access to the rest of the map. The best maps in f_{exp} have bases far from each other, with impassable regions between them to make navigation more difficult.

4.2 Optimizing Multiple Fitness Dimensions

Experiments in Section 4.1 showed that maps optimized for a single dimension usually have interesting traits, but lack the necessary features needed for competitive play. Combining multiple fitness dimensions into a weighted sum and using it as the objective function for the genetic algorithm is expected to generate better designed maps. Experiments in this section will assess the combined optimization of two or more fitness dimensions. For space considerations, the following representative fitness function combinations are explored in this paper:

- $F_{res} = \frac{1}{2} f_{res} + \frac{1}{2} b_{res}$
- $F_{saf} = \frac{1}{2} f_{saf} + \frac{1}{2} b_{saf}$
- $F_{exp} = \frac{1}{2} f_{exp} + \frac{1}{2} b_{exp}$
- $F_{all-f} = \frac{1}{3} f_{res} + \frac{1}{3} f_{saf} + \frac{1}{3} f_{exp}$
- $F_{all-b} = \frac{1}{3} b_{res} + \frac{1}{3} b_{saf} + \frac{1}{3} b_{exp}$
- $F_{all} = \frac{1}{6} f_{res} + \frac{1}{6} f_{saf} + \frac{1}{6} f_{exp} + \frac{1}{6} b_{res} + \frac{1}{6} b_{saf} + \frac{1}{6} b_{exp}$

Figure 5 shows the evolutionary progress of the contributing fitness scores for the fittest individuals in the different fitness combinations; displayed values are averaged from 20 independent runs, with the standard deviation values depicted as error bars. The highest scoring final individual among the 20 runs for each fitness combination is shown

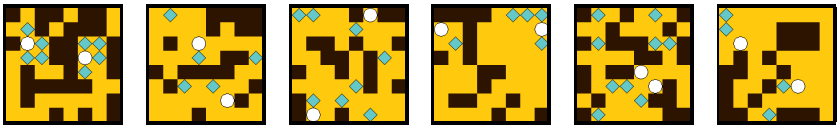


(a) F_{res} (b) F_{saf} (c) F_{exp}

(d) F_{all-f} (e) F_{all-b} (f) F_{all}

Fig. 5. Optimization of the contributing fitness dimensions in the fittest individuals in the population, when evolving strategy maps for multiple fitness dimensions. Error bars represent standard deviation across 20 runs.

in Fig. 6. Results indicate that, for F_{res}, F_{saf} and F_{exp}, optimization is dominated by the fitness dimension of balance, which is optimized quickly and largely determines the selection of parents; since f_{res} and f_{saf} are slower to optimize, they do not achieve as high scores as when optimized individually. Their best maps in Fig. 6 are, however, of good quality: F_{res} has an equal number of resources adjacent to each base, while F_{saf} has a single chokepoint in the map, with areas on either side of the chokepoint being of equal size; finally, F_{exp} has bases far away from each other, hidden behind large impassable regions. On the more complex fitness combinations, F_{all-b} easily finds optimal maps in all the contributing fitnesses; F_{all-f}, on the other hand, does not achieve as high scores as when each dimension is optimized on its own, but it does not seem to be dominated by any dimension. Finally, the optimization of F_{all} unsurprisingly shows difficulties in reaching high fitness scores in all contributing fitnesses; fitness dimensions of balance dominate f_{res}, f_{saf} and f_{exp}, which are harder to optimize and show higher sensitivity with respect to their convergence (as depicted by their large standard deviation values). The best maps for F_{all-f} have all the features of the contributing fitness dimensions, but are very unbalanced. The best maps for F_{all-b} have bases adjacent to each other, since that is often optimal both for b_{exp} and b_{saf}; such maps are not generally playable in a strategy game. Despite the slow and asymmetrical optimization of F_{all}, its best maps are probably the most appropriate for use in a strategy game. Even better maps may be possible with other combinations of criteria, such as minimizing f_{saf} and f_{res} to create maps suited for aggressive gameplay.



(a) F_{res} (b) F_{saf} (c) F_{exp} (d) F_{all-f} (e) F_{all-b} (f) F_{all}

Fig. 6. Best final individuals among 20 evolutionary runs, optimized for a combination of two or more fitness dimensions displayed in each map's caption

5 Conclusion

This paper presented the concept of map sketches which are appropriate for a mixed-initiative tool allowing for the collaborative creation of maps for strategy games. With this tool, a computer can evaluate the human-authored map in real-time, and the user can request improved maps in several fitness dimensions. Since maps are represented as low-resolution sketches, computational time for map evaluation and optimization is minimized, while designer fatigue and fixation are also expected to be reduced. Experiments conducted on random initial populations, with sufficient time to evolve, demonstrated that small two-player maps of high quality can be easily optimized by the genetic algorithm on one or more fitness dimensions. While the combination of more fitness dimensions into a weighted sum limited the efficiency of optimization somewhat, the evolved maps are more appropriate for use in strategy games than those optimized on a single fitness dimension. Aggregating multiple objectives into a single fitness has its caveats, although the fitness dimensions combined in this paper are not particularly conflicting. While multi-objective evolutionary algorithms such as those used in [12] could potentially lead to better results, such algorithms are generally more computationally demanding and thus inappropriate for the fast feedback mechanisms of a mixed-initiative tool. Preliminary tests show that, while slower to evolve, the same processes can optimize, to a high quality, larger maps with more bases and resources.

Future work includes using the evolutionary methods described in this paper to optimize a user-created map via the map editor. The experiments presented in this paper were strictly offline, starting from large random populations and without any time constraint. Online evolution while a user edits the map may have its own challenges, particularly since the general form of the user-created map must be retained. Additionally, while the fitness scores are fast to calculate, they are built on design decisions that may not be accurate enough: for instance, exploration is measured via flood fill (which is not how players explore maps in strategy games), while the safety metric underestimates the impact of chokepoints. Future work should refine the existing fitnesses and possibly add new ones; however, the more fitnesses are added, the more difficult their simultaneous optimization will become. Finally, while the user may enjoy controlling which fitness dimension is being optimized, it may also become cumbersome and unintuitive to guess which combination of fitness dimensions are needed for the designer's purposes. This can be addressed by indirectly modelling the designer's intentions based on their history of content authoring and content selection through *choice-based interactive evolution*, which has been used in previous work to automatically adapt models of aesthetic preferences [7].

Acknowledgements. The research is supported, in part, by the FP7 ICT project SIREN (project no: 258453) and by the FP7 ICT project C2Learn (project no: 318480).

References

1. Bjork, S., Holopainen, J.: Patterns in Game Design. Charles River Media (2004)
2. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games 2(1), 1–16 (2010)
3. Frade, M., de Vega, F.F., Cotta, C.: Evolution of Artificial Terrains for Video Games Based on Accessibility. 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, Part I. LNCS, vol. 6024, pp. 90–99. Springer, Heidelberg (2010)
4. Hastings, E.J., Guha, R.K., Stanley, K.O.: Evolving content in the galactic arms race video game. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 241–248 (2009)
5. Kimbrough, S.O., Koehler, G.J., Lu, M., Wood, D.H.: On a feasible-infeasible two-population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. European Journal of Operational Research 190(2), 310–327 (2008)
6. Liapis, A., Yannakakis, G.N., Togelius, J.: Neuroevolutionary constrained optimization for content creation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games, pp. 71–78 (2011)
7. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games 4(3), 213–228 (2012)
8. Liapis, A., Yannakakis, G.N., Togelius, J.: Limitations of choice-based interactive evolution for game level design. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (2012)
9. Schoenauer, M., Michalewicz, Z.: Evolutionary Computation at the Edge of Feasibility. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 245–254. Springer, Heidelberg (1996)
10. Smith, A., Mateas, M.: Answer set programming for procedural content generation: A design space approach. IEEE Transactions on Computational Intelligence and AI in Games 3(3), 187–200 (2011)
11. Sorenson, N., Pasquier, P., DiPaola, S.: A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games 3(3), 229–244 (2011)
12. Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelback, J., Yannakakis, G.N.: Multiobjective exploration of the StarCraft map space. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 265–272 (2010)
13. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games (99), 172–186 (2011)
14. Yannakakis, G.N.: Game AI revisited. In: Proceedings of the ACM Computing Frontiers Conference, pp. 285–292 (2012)

A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars

Raúl Lara-Cabrera, Carlos Cotta, and Antonio J. Fernández-Leiva

Department "Lenguajes y Ciencias de la Computación", ETSI Informática, University of Málaga, Campus de Teatinos, 29071 Málaga – Spain
{raul,ccottap,afdez}@lcc.uma.es

Abstract. Procedural content generation (PCG) is the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible gameplay spaces. This methodology brings many advantages to game developers, such as reduced memory consumption. This works presents a procedural balanced map generator for a real-time strategy game: *Planet Wars*. This generator uses an evolutionary strategy for generating and evolving maps and a tournament system for evaluating the quality of these maps in terms of their balance. We have run several experiments obtaining a set of playable and balanced maps.

1 Introduction

Procedural content generation (PCG) refers to creating game content automatically, through algorithmic means. This content refers to all aspects of the game that affect gameplay other than non-player characters (NPCs), such as maps, levels, dialogues, characters, rule-sets and weapons. PCG is interesting for the game developing community due to several reasons, such as reduced memory consumption and the saving in the expense of manually creating game content.

Due to the benefits detailed previously, procedural content generation has been used in many well-known videogames. *Borderlands* [10] uses a PCG system to create weapons and items, which can alter their firepower, rate of fire, and accuracy, add in elemental effects such as a chance to set foes on fire or cover them in burning acid, and at rare times other special bonuses such as regenerating the player's ammo. PCG system is also used to create the characteristic of random enemies that the player may face. Another example of a game that uses PCG is *Minecraft* [15], a sandbox-building game with an infinite map which is expanded dynamically. *Spore* [14] is a god game simulation that contains multiple levels of play, from starting as a multi-celled organism in a tide pool, up to exploring a dynamically generated universe with advanced UFO technology. The music of the game is also procedurally generated.

From an academic point of view, there are several papers related to procedural map generation. In [19] the authors designed a system for offline/online generation of tracks for a simple racing game. A racing track is created from a parameter

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 274–283, 2013.
© Springer-Verlag Berlin Heidelberg 2013

vector using a deterministic genotype-to-phenotype mapping. A search-based procedural content generation (SBPCG) algorithm for strategy game maps is proposed in [21] from a multi-objective perspective. A multi-objective evolutionary algorithm is used for searching the space of maps for candidates that satisfy pairs of these multiple objectives. Another search-based method for generating maps is presented in [20]. In this case, the maps are generated for the game *Starcraft* [2]. Frade et al. have introduced the idea of terrain programming, namely the use of genetic programming to evolve playing maps for videogames, using either subjective human-based feedback [7], [8] or automated quality measures such as accessibility [6] or edge-length [9]. In [13] the authors describe a search-based map generator for an abstract version of the real-time strategy game *Dune 2*. Map genotypes are represented as low-resolution matrices, which are then converted to higher-resolution maps through a stochastic process involving cellular automata.

Real-time strategy (RTS) games are a genre of videogames which require managing different kind of units and resources in real-time. In a RTS game the participants position and maneuver units and structures under their control to secure areas of the map and/or destroy their opponents' assets. In a typical RTS, it is possible to create additional units and structures during the course of a game, but this is generally limited by the number of accumulated resources. These resources are gathered by controlling special points on the map and/or possessing certain types of units and structures devoted to this purpose. The typical game of the RTS genre features resource gathering, base building, in-game technological development and indirect control of units. They are usually played by two or more players (human or not). These players have to deal with incomplete information during the game (the map is covered by fog of war, the technology developed by a player is unknown by every other player, ...). These features make RTS games a great tool for computational intelligence research, since a RTS game player needs to master many challenging problems such as resource allocation [3,11], strategy planning [1,5,16] and opponent's strategy prediction [4,18]. In addition, procedural content generation can be used to create maps, units and technologies for RTS games. Traditionally, academic game artificial intelligence (AI) was mainly linked to non player character (NPC) behavior and pathfinding. However, there are new research areas that have recently provided innovative solutions for a number of game development challenges, like player experience modeling (PEM), procedural content generation (PCG) and large scale game data mining [23].

This paper introduces a map generation method for a RTS game that can be categorized (using the taxonomy proposed in [22]) as an off-line method that generates necessary content, using random seeds and deterministic generation and following a generate-and-test schema. This method generates balanced maps, i.e. maps where players do not have any advantage over their opponents regardless of their ability or strategy type.

2 Game Description

Planet Wars is a real-time strategy (RTS) game based on *Galcon* and used in the *Google AI Challenge 2010*. The game is set in outer space and its objective is to take over all the planets on the map or eliminate all of your opponents ships. A game of Planet Wars takes place on a map that contains several planets with some number of ships on it. Each planet may have a different number of ships. The planets may belong to some player or may be neutral. The game has a certain maximum number of turns and it may end earlier if one of the players loses all his ships, and in this case the player that has ships remaining wins instantly. If both players have the same number of ships when the game ends, it is considered a draw. On each turn, the player may choose to send fleets of ships from any planet he owns to any other planet on the map. He may send as many fleets as he wishes on a single turn as long as he has enough ships to supply them. After sending fleets, each planet owned by a player (not owned by neutral) will increase the forces there according to that planets growth rate. Different planets have different growth rates. The fleets will then take some number of turns to reach their destination planets, where they will then fight those opposing forces there and, if they win, take ownership of the planet. Fleets cannot be redirected during travel. Players may continue to send more fleets on later turns even while older fleets are in transit. Despite players make their orders on a turn-by-turn basis, they issue these orders at the same time, so we can treat this game as a real-time game.

Maps have no particular dimensions and are defined completely in terms of the planets and fleets in them. They are defined in plain text files, with each line representing a planet or a fleet. Planet positions are specified relative to a common origin in Euclidean space. The coordinates are given as floating point numbers. Planets never move and are never added or removed as the game progresses. Planets are not allowed to occupy the exact same position on the map. A planet can be neutral or owned by some player. The number of ships is given as an integer, and it may change throughout the game. Finally, the growth rate of the planet is the number of ships added to the planet after each turn. It is given as an integer and it also represents the size (i.e. radius) of the planet. If the planet is currently owned by neutral, the growth rate is not applied. Only players can get new ships through growth. The growth rate of a planet will never change.

3 A Procedural Balanced Map Generator

A map is balanced if players do not have any advantage over their opponents regardless of their ability or strategy type. Due to this feature, this kind of maps are important for the evaluation of human or artificial players, since they do not boost the performance of any player. In order to create balanced maps, we have designed a procedural map generator that is composed of an evolutionary strategy and a tournament system. The evolutionary strategy is responsible for

generating new random maps and evolving them, while the tournament system evaluates the quality of the generated maps based on the results obtained from several matches between non-player characters.

3.1 Evolutionary Strategy

As mentioned before, the evolutionary strategy (ES) is devoted to generate maps with an arbitrary number of neutral planets ranged between 15 and 30 following the rules of the game. These maps are the individuals of the ES and they are represented by a variable-length vector of planets. As described on the previous section, every planet has five properties: x-position, y-position, owner, growth rate and number of ships. We have fixed planet's holders so that players own the first and second planet of every map while the rest of the planets are neutral, so individuals' genes are groups of 4 parameters. In addition to these parameters the algorithm needs 4 additional parameters since this is a self-adaptive evolutionary strategy so the parameters of the mutation operator evolve along with the planets' parameters.

Regarding these planets' parameters, two of them have real values (x and y position) while the other two (growth rate and number of ships) have integer values. In addition to this, x and y positions range between 0 and 15, while the growth rate and the number of ships fluctuate between 1 and 5, and 100.

Due to the types of the parameters (real and integer), the evolutionary strategy uses an hybrid mutation operator that uses different mutation methods for real and integer parameters. The operator mutates x and y coordinates following a Gaussian mutation scheme with self-adaptive step sizes. The problem with applying Gaussian mutation to integer values is that this kind of mutation generates real-value perturbations which are rounded to an integer perturbation. To prevent this, this mutation operator uses a method [12,17] that generates suitable integer mutations for the growth rate and number of ships. This method is similar to the self-adaptive mutation of real values, with a set of step-size parameters controlling the strength of the mutation, but using the difference of two geometrically distributed random variables to generate the perturbation instead of the normal distributed random variables used by the real values method.

In the case of real-valued parameters $\langle x_1, ..., x_n \rangle$ they are extended with n step sizes, one for each parameter, resulting in $\langle x_1, ..., x_n, \sigma_1, ..., \sigma_n \rangle$. The mutation mechanism is specified as follows:

$$\sigma_i' = \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)}$$
$$x_i' = x_i + \sigma_i \cdot N_i(0,1)$$

where $\tau' \propto 1/\sqrt{2n}$, and $\tau \propto 1/\sqrt{2\sqrt{n}}$. A boundary rule is applied to step sizes to prevent standard deviations very close to zero: $\sigma_i' < \epsilon_0 \Rightarrow \sigma_i' = \epsilon_0$ (in this algorithm, σ_0 represents 1% of the parameter's range).

Regarding integer-valued parameters $\langle z_1, ..., z_m \rangle$ they are extended in a similar way than real-valued parameters, resulting in $\langle z_1, ..., z_m, \varsigma_1, ..., \varsigma_m \rangle$. The mutation mechanism is specified as follows:

$$\varsigma_i' = \max(1, \varsigma_i \cdot e^{\tau \cdot N(0,1) + \tau' \cdot N(0,1)})$$

$$\psi_i = 1 - (\varsigma_i'/m) \left(1 + \sqrt{1 + \left(\frac{\varsigma_i'}{m} \right)^2} \right)^{-1}$$

$$z_i' = z_i + \left\lfloor \frac{ln(1 - U(0,1))}{ln(1 - \psi_i)} \right\rfloor - \left\lfloor \frac{ln(1 - U(0,1))}{ln(1 - \psi_i)} \right\rfloor$$

where $\tau = 1/\sqrt{2m}$ and $\tau' = 1/\sqrt{2\sqrt{m}}$. As described before, the main difference between the two methods is the distribution used to generate the perturbation.

Continuing with operators, this evolutionary strategy uses a "cut and splice" operator that recombines two individuals by swapping cut pieces with different sizes (this way, generated maps have different numbers of planets). We have chosen this operator due to the arbitrary length of the individuals. Table 1 summarizes the algorithm's parameters.

Table 1. Algorithm's parameters

Representation	Vector of planets
Recombination	Cut and slice
Mutation	Gaussian perturbation (real) and geometric difference (integers)
Parent selection	Binary tournament
Survivor selection	$(\mu + \lambda)$ with $\mu = 10$ and $\lambda = 100$
Speciality	Self-adaption of mutation step sizes and genome length

To evaluate the quality of every individual the algorithm runs a tournament that takes place on the generated map between several players. Once the tournament has finished, the algorithm gathers the individual's fitness from the result of the tournament. Equation (3) defines the fitness, with N_m being the number of matches played during the tournament, t_i being the number of turns of match i, K_i being the added up percentage of occupied planets by both players at the end of the game, $P_{ij}^{(1)}$, $P_{ij}^{(2)}$ being the percentage of owned planets by player 1 and player 2 respectively, in match i and turn j and $S_{ij}^{(1)}$, $S_{ij}^{(2)}$ being the percentage of the total ships owned by player 1 and player 2 respectively in match i and turn j.

$$\bar{P}_i = \frac{\sum_{j=1}^{t_i} \left| P_{ij}^{(1)} - P_{ij}^{(2)} \right|}{t_i} \tag{1}$$

$$\bar{S}_i = \frac{\sum_{j=1}^{t_i} \left| S_{ij}^{(1)} - S_{ij}^{(2)} \right|}{t_i} \tag{2}$$

$$fitness = \left(\frac{1}{N_m} \sum_{i=1}^{N_m} \frac{K_i \cdot t_i}{\bar{P}_i + \bar{S}_i + 1} \right)^2 \quad (3)$$

Fitness function (3) promotes balanced maps through its components: $\bar{P}_i$ and $\bar{S}_i$ promotes maps where players have similar number of planets and ships (it sums up 1 to avoid dividing by zero), while t_i promotes long games because it means that there have not been a winner or the winner is determined nearly at the end of the game. Finally, K_i promotes maps where there have been high activity, i.e. players have conquered many planets.

3.2 Tournament System

The tournament system is the component devoted to evaluate the quality of the generated maps. This component runs a set of *Planet Wars* games between an arbitrary number of non-player characters (NPC). Every NPC plays at least a game against each other, although this parameter is customizable. The tournament system evaluates every game analyzing the logs generated by a Java console-style tool, which was developed by *Google* for the *Google AI Challenge 2010*. The evolutionary strategy provides the maps to the tournament system, which evaluates the map and returns this evaluation to the former.

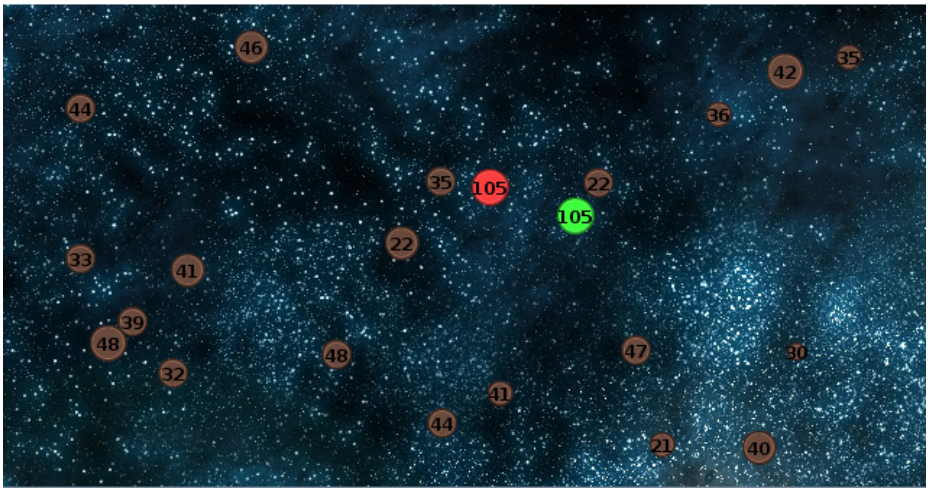


Fig. 1. An example of a balanced procedural generated map

4 Experiments and Results

We have run two experiments (10 executions each) with different parameters, obtaining a set of playable and balanced maps (one of these maps is shown in

Figure 1). The first experiment uses an evolutionary strategy with the parameters described before (see Table 1), using a self-adapting strategy for mutation steps and genome length (i.e. number of planets in the map), while the second experiment uses the same parameters except for the fixed genome length (23 planets in every map since the number of planets ranges between 15 and 30). We have evaluated the quality of the maps using the tournament system with three NPCs who were participants of the *Google AI Challenge 2010* (*Manwe*[1], *Flagscapper's bot*[2] and *fglider's bot*[3]) , all of them ranked in the top 100 and having their source code available (there were over 4600 participants). The maximum number of turns per game has been limited to 400 turns. We have observed that the planets of many generated maps are much separated from each other. Maps of this kind should be considered as balanced maps because it takes a long time (number of turns) to reach the enemy and fight him, so players can conquest new planets without troubles and their fleet grows with a similar rate —the fitness of this kind of maps will be high because of the low difference between the number of owned planets and ships.

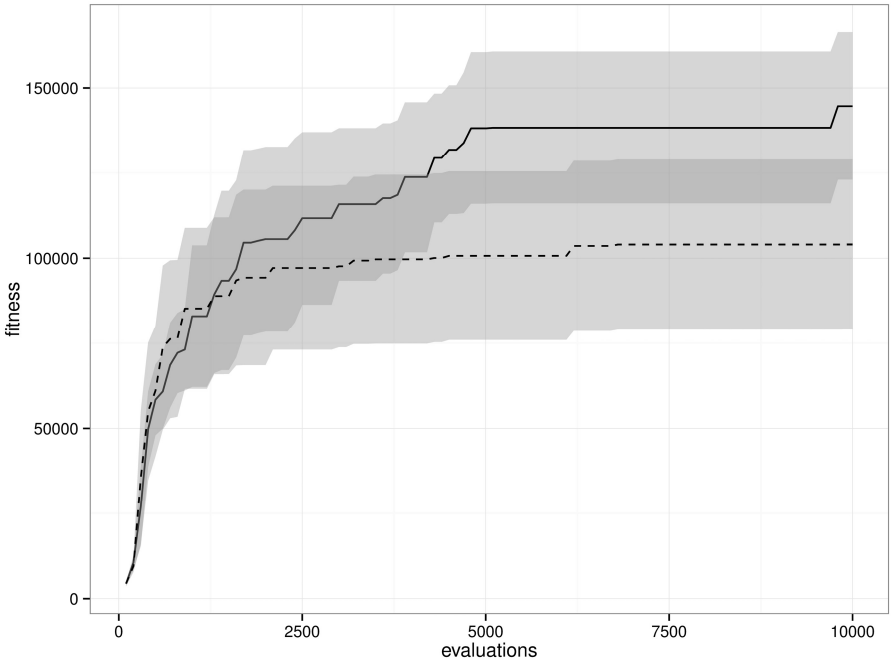


Fig. 2. Evolution of the averaged fitness

[1] `https://github.com/Manwe56/Manwe56-ai-contest-planet-wars`
[2] `http://flagcapper.com/?c1`
[3] `http://planetwars.aichallenge.org/profile.php?user_id=8490`

Figure 2 shows the evolution of the averaged fitness for the two experiments (solid line for self-adaption of the genome size and mutation steps and dashed line for self-adaption of mutation steps only). Grey areas show the standard mean error of the averaged fitness values. As we can see in the figure, both experiments have a similar behavior over the first evaluations but the self-adaptive algorithm (experiment 1) gets a better fitness over the subsequent evaluations. Figure 3 shows the evolution of the averaged number of planets in the best map (i.e. the individual with the highest fitness). As we can observe in the figure, after some evaluations, this number converges to the value 17, so we should think that maps with 17 planets are more balanced than other maps with a higher number of planets.

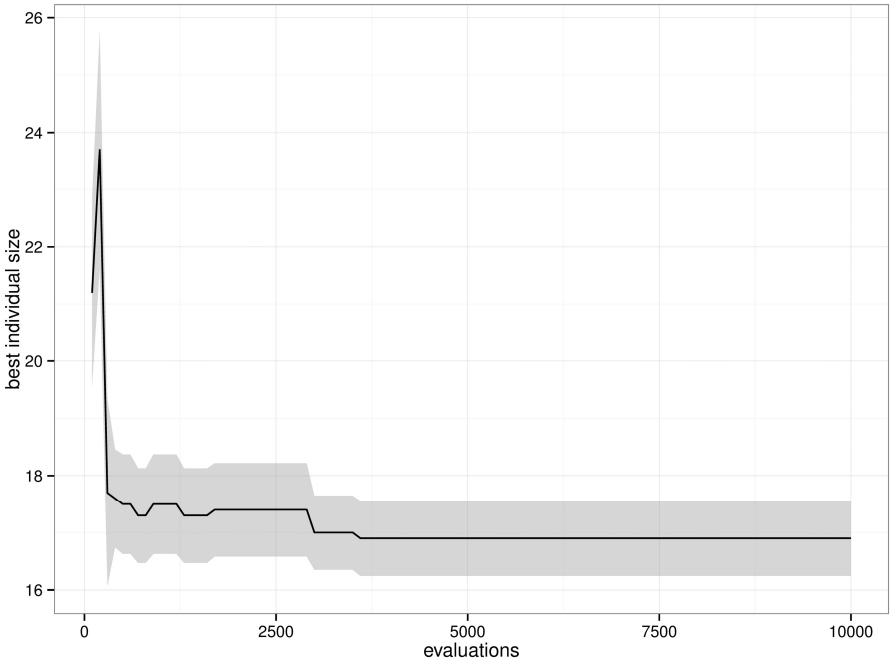


Fig. 3. Evolution of the averaged number of planets in the best map

5 Conclusion and Future Work

In this paper we have introduced a procedural map generator for a RTS game that is capable of generating balanced maps for a real-time strategy game: *Planet Wars*. These maps do not give any advantage to the players regardless of their ability or strategy type. This generator turns *Planet Wars* into an endless game and makes the game more interesting to weak players (since they do not lose with ease), raising the competitiveness of the stronger player with harder challenges.

Despite this algorithm generates fully playable maps, there are some improvements that could be made to this generator. The evolutionary strategy uses only a mutation operator, so it could be interesting to improve the breeding pipeline, adding additional or improved mutation and recombination operators. Moreover, maps generated by this algorithm are not symmetrical and some planets should be overlapped, although the evolutionary strategy avoids overlapped planets since this is an advantage to the player who has this overlapped planets nearer. In addition to this, it is possible to obtain other characteristics of the maps that make them more balanced, such as the averaged distance between the planets, players' initial positions or the distribution of the planets over the map.

In the near future, we are going to introduce interactivity and pro-activity to this procedural map generator, in order to improve its performance and the quality of generated maps.

Acknowledgements. This work is partially supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project P10-TIC-6083 (DNEMESIS).

References

1. Aha, D.W., Molineaux, M., Ponsen, M.: Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 5–20. Springer, Heidelberg (2005)
2. Blizzard Entertainment: Starcraft. Blizzard Entertainment (1998)
3. Chan, H., Fern, A., Ray, S., Wilson, N., Ventura, C.: Online planning for resource production in real-time strategy games. In: Boddy, M.S., et al. (eds.) International Conference on Automated Planning and Scheduling, pp. 65–72. The AAAI Press (2007)
4. Cheng, D., Thawonmas, R.: Case-based plan recognition for real-time strategy games. In: El-Rhalibi, A., van Welden, D. (eds.) GameOn Conference, pp. 36–40. EUROSIS (2004)
5. Chung, M., Buro, M., Schaeffer, J.: Monte Carlo Planning in RTS Games. In: IEEE Symposium on Computational Intelligence and Games. IEEE (2005)
6. Frade, M., de Vega, F.F., Cotta, C.: Evolution of Artificial Terrains for Video Games Based on Accessibility. 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, Part I. LNCS, vol. 6024, pp. 90–99. Springer, Heidelberg (2010)
7. Frade, M., de Vega, F.F., Cotta, C.: Modelling Video Games' Landscapes by Means of Genetic Terrain Programming - A New Approach for Improving Users' Experience. 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. 485–490. Springer, Heidelberg (2008)
8. Frade, M., de Vega, F.F., Cotta, C.: Breeding terrains with genetic terrain programming: The evolution of terrain generators. International Journal of Computer Games Technology 2009 (2009)

9. Frade, M., de Vega, F.F., Cotta, C.: Evolution of artificial terrains for video games based on obstacles edge length. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2010)
10. Gearbox Software: Borderlands. 2K Games (2009)
11. Kovarsky, A., Buro, M.: A First Look at Build-Order Optimization in Real-Time Strategy Games. In: Wolf, L., Magnor, M. (eds.) GameOn Conference, pp. 18–22. EUROSIS (2006)
12. Li, R.: Mixed-integer evolution strategies for parameter optimization and their applications to medical image analysis. Ph.D. thesis (2009)
13. Mahlmann, T., Togelius, J., Yannakakis, G.N.: Spicing Up Map Generation. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 224–233. Springer, Heidelberg (2012)
14. Maxis: Spore. Electronic Arts (2008)
15. Mojang: Minecraft. Mojang (2011)
16. Ng, P.H.F., Li, Y.J., Shiu, S.C.K.: Unit formation planning in RTS game by using potential field and fuzzy integral. In: Fuzzy Systems, pp. 178–184. IEEE (2011)
17. Rudolph, G.: An Evolutionary Algorithm for Integer Programming. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 139–148. Springer, Heidelberg (1994)
18. Synnaeve, G., Bessiere, P.: A bayesian model for opening prediction in RTS games with application to StarCraft. In: Computational Intelligence and Games, pp. 281–288. IEEE (2011)
19. Togelius, J., De Nardi, R., Lucas, S.: Towards automatic personalised content creation for racing games. In: IEEE Symposium on Computational Intelligence and Games, CIG 2007, pp. 252–259 (2007)
20. Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelback, J., Yannakakis, G.: Multiobjective exploration of the starcraft map space. In: 2010 IEEE Symposium on Computational Intelligence and Games (CIG), pp. 265–272 (2010)
21. Togelius, J., Preuss, M., Yannakakis, G.N.: Towards multiobjective procedural map generation. In: Proceedings of the 2010 Workshop on Procedural Content Generation in Games, pp. 1–8 (2010)
22. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games 3(3), 172–186 (2011)
23. Yannakakis, G.N.: Game ai revisited. In: Proceedings of the 9th Conference on Computing Frontiers, CF 2012, pp. 285–292. ACM, New York (2012)

Mechanic Miner: Reflection-Driven Game Mechanic Discovery and Level Design

Michael Cook, Simon Colton, Azalea Raad, and Jeremy Gow

Computational Creativity Group
Imperial College, London

Abstract. We introduce Mechanic Miner, an evolutionary system for discovering simple two-state game mechanics for puzzle platform games. We demonstrate how a reflection-driven generation technique can use a simulation of gameplay to select good mechanics, and how the simulation-driven process can be inverted to produce challenging levels specific to a generated mechanic. We give examples of levels and mechanics generated by the system, summarise a small pilot study conducted with example levels and mechanics, and point to further applications of the technique, including applications to automated game design.

Keywords: game mechanics, automated game design, platform games.

1 Introduction

Procedural content generation (PCG) is a highly active area of research, both in academia and game development. The ability to generate high-quality content on-demand and in large volumes not only can improve a game's quality, but has made possible new types of game dependent on their ability to generate content in this way - such as Betts' *In Ruins* or Smith's *Endless Web* [5]. However, much PCG focuses on the generation of consumable data, such as terrain, quests, items or narrative. In order to explore the concept of fully-automated game design, we must find techniques for generating all aspects of a game, including higher-level concepts that describe how game systems interact with one another; and in particular how they interact with the player.

Game mechanics are an important type of player-game interactions; they range from the well-known (such as jumping in a platform game) to the experimental (such as in [9], where the player's movement was controlled by shouting loudly or softly into a microphone). Novel mechanics are still a major source of interest in the independent development community, with many events focusing on mechanics[1] and many awards rewarding innovative design[2]. One approach to generating game mechanics is to compose them out of smaller rules or concepts defined by hand. However, doing this explores a restricted search space, and limits the system's potential for novelty or surprise. Reducing the use of predefined rules or domain knowledge may help mitigate this.

[1] E.g. The Experimental Gameplay Project (`http://experimentalgameplay.com/`).
[2] E.g. The Independent Games Festival's *Nuovo Award*.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 284–293, 2013.
© Springer-Verlag Berlin Heidelberg 2013

We introduce here Mechanic Miner, an evolutionary system designed to generate simple game mechanics, and then design levels that specifically require the use of those mechanics in their solution. We show how the reflective properties of a programming language can be used to generate mechanics programmatically without metalevel domain knowledge, and demonstrate a simulation-driven approach to evaluating mechanics for utility. We then show how, by inverting the evolutionary system, we can use the same process and principles to design levels for specific mechanics, with control over features like difficulty and complexity.

The rest of the paper is organised as follows: in section 2 we describe the process of evolving game mechanics through simulation; in section 3 we show how this process can be reversed to design levels that use certain mechanics; in section 4 we give examples of generated content and describe a small pilot study; in section 5 we review related work to this project and distinguish our approach and in section 6 we conclude and describe some directions for future work.

2 Automatic Generation of Game Mechanics

2.1 Background

Reflection. Reflection is the ability of a programming language to inspect itself at runtime, allowing for the runtime creation of new classes, the modification of code, and the inspection, invocation and alteration of fields and methods. The following code retrieves a list of objects describing the fields of an object *o*:

```
Class<?> c = o.getClass();  List<Field> fs = c.getFields();
```

Mechanic Miner is further extended by the open source Reflections library[3] in order to overcome the limitations of Java's standard reflection.

Toggleable Game Mechanics. In the remainder of this paper we refer to *toggleable game mechanics* (TGM), an intentionally simplified subspace of game mechanics that we identify for the purposes of demonstrating Mechanic Miner. A TGM is an action the player can take to change the state of a variable. That is, given a field foo and a modification function m with inverse m^{-1}, a TGM is an action the player can take which applies $m(foo)$ when pressed the first time, and $m^{-1}(foo)$ when pressed a second time. The action may not be perfectly reversible; if foo is changed elsewhere in the code between the player taking actions m and m^{-1}, the inverse may not set foo back to the value it had when m was applied to it. For instance, if foo is the player's x co-ordinate, the and the player moves around after applying m, then their x co-ordinate will not return to its original value after applying $m{-}1$, as it was modified by the player moving.

Flixel and Flixel-Android. Flixel[4] is a popular open-source game library built on top of Actionscript 3. Flixel-Android[5] is a port of the Flixel library to

[3] http://code.google.com/p/reflections/
[4] http://www.flixel.org
[5] http://code.google.com/p/flixel-android/

the LibGDX framework. It follows the same structure and has the same method calls (with some concessions made for differences in programming language capabilities). LibGDX[6] is a Java library that compiles to many platforms.

2.2 Mechanic Miner

Mechanic Miner is an evolutionary system for searching a codebase for *usable* TGMs. The codebase is defined as any code in the game being analysed – for our experiments, this means both the Flixel-Android library code, and the code of a simple platform game written using the library. We define a *usable* mechanic as one which enables the player to overcome some obstacle in order to complete a task, such as reaching an exit. Figure 1 shows a sample level where the player's progress to the exit, marked 'X', is blocked by a tower of impassable blocks. The player starts in the square marked 'S', falls to the ground under gravity, and is unable to jump over the tower. A usable TGM is any TGM that allows them to reach the exit from this starting configuration. One example would be a mechanic that toggled gravity off, allowing the player to jump as high as they wanted temporarily, and thereby leap over the tower in the centre.

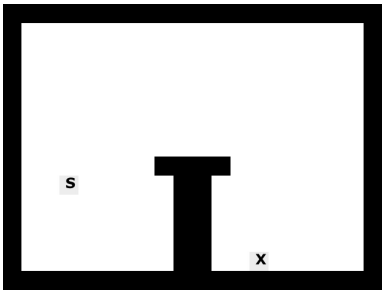


Fig. 1. A sample puzzle level. The player starts at the 'S' mark, and must reach the 'X' mark. Black squares are solid, and gravity acts on the player pulling them down.

Population Generation via Codebase Inspection. In section 2.1, we defined a TGM as a combination of a game variable and a function that modifies it. In Mechanic Miner, we formalise this by describing a mechanic as being comprised of two parts: a `java.lang.Field` object that refers to a field present either in the game library's code or the code of the template game; and a `Modifier` that encapsulates an operation on a field to change it. In order to generate such mechanics, we use reflection to search through all the classes defined in the Flixel-Android library and the template game. We choose fields at random, and based on their type we randomly select a modifier to use. The two together comprise a TGM. Modifiers are type-specific because the range of operations that can be performed on a field are restricted by the type system. For numerical

[6] `https://github.com/libgdx/libgdx`

types like `float` or `integer`, we offer modifiers that double or halve the value, or multiply it by -1. For `boolean` fields, we invert the sign of the boolean. The set of modifiers we use is limited for this work to simplify the system.

Fitness Evaluation via Simulation. We evaluate the fitness of a mechanic during evolution by simulating its use in the sample level, testing its usability, and calculate its score based on the area the player gains access to. The process of simulation is intended to find all valid sequences of *actions* under a certain length, using a breadth first search. An action is a set of steps that the simulated character performs until it can no longer reach any new space. For instance, the action `MOVE` encapsulates moving left or right, until all areas reached by only moving left or right have been found.

The breadth-first search process simulates one action at a time, and for each level location reached by the character the simulator checks a data structure for that location to see if the same (or shorter) sequence of actions has already reached that location. If not, the sequence is new for this co-ordinate, and so it adds a new node to the search space to be explored. Because this node is one move longer than the node that was just explored, the breadth-first search will only expand it after it has finished searching all the nodes with shorter sequences. In this way, we find the shortest sequence that reaches the exit. For the mechanics and levels described in this paper, the actions used were `MOVELEFT`, `MOVERIGHT`, `JUMP`, `SPECIAL` and `NOTHING`. `NOTHING` is an action used when simulating a fall.

When a sequence of moves that reaches the exit location has been found, it is returned instantly (as this proves the TGM is *usable*, under the earlier definition). The fitness of a mechanic in this instance is proportional to the amount of the game world that the player was able to access before reaching the exit. This is a good metric as it allows the system to avoid finding mechanics that are too empowering (allowing the player to reach everywhere in the game world) or not empowering enough (mechanics that, for instance, instantly teleport the player to the exit location, which offer little flexibility for interesting level design).

Crossover, Mutation, and Evolutionary Parameters. Mutation of a TGM is performed by either randomly varying the modifier on the mechanic (e.g. changing a `Float`'s modifier from *Halve* to *InvertSign*), or by changing the TGM's target field to another field from the same class within the game engine. Specific mutations are selected randomly from all legal mutations for the mechanic (if a type has only one modifier, for instance, it will not attempt to randomly reassign the modifier as it will result in an identical mechanic). Crossover of two mechanics uses uniform crossover where the two mechanics affect fields of the same type. In the case that the two mechanics do not, mutation takes place instead. 10% of the new population is comprised of newly-generated TGMs.

A standard run of Mechanic Miner maintains a population of 100 mechanics, evolved for 15 generations. Simulation (for fitness evaluation) is limited to ten discrete actions, such as `MOVERIGHT`, before the simulator stops under the assumption that no solution can be found (the simplicity of the sample problem

means that it is unlikely that solutions longer than ten moves will be found - none found during experimentation took more than ten moves). These parameters were all determined through experimentation with the system.

3 Mechanic-Led Level Design

Once a TGM has been identified by Mechanic Miner as potentially usable for the candidate problem, we can use it as the basis for designing levels that specifically require the use of that mechanic by players en route to finishing the level. This section outlines the process of evolving level designs for specific mechanics.

Population Generation. A level is described abstractly as a collection of geometric shapes that prescribe the placement of impassable blocks, similar to the system described in [1]. Each shape is either a *Line* or a *Box*. Boxes may be filled or outlines only, and Lines may be standard blocks, or spikes which kill the player on contact. The generation of a level consists of the random placement of shapes, with a limit on the minimum and maximum number of shapes a level can contain. For the levels generated in this paper, they contain no fewer than 2 elements and no more than 20.

Simulation-Driven Fitness Evaluation. In order to evaluate the level, we simulate multiple attempts at solving it, using the simulation method outlined in the previous section. The first simulation does not use any mechanics, to test if the level can be completed without mechanic use at all. If this is the case, the level receives a penalty to its fitness, as we want to end up with levels that can only be solved using the new mechanic. The second simulation is a standard simulation using the mechanic found by Mechanic Miner. A third simulation can also be run in order to evaluate the difficulty of the level. In this simulation, the mechanic is used but the amount of time processed between each simulation step is increased. By increasing the time the game engine runs between actions by the simulator, we can simulate the player having slower reaction times. This makes it possible to approximately parameterise the difficulty of a level, by penalising the fitness of levels that can be solved with long reaction times.

When the three simulations are complete, assuming the level was not completable without mechanic use, the fitness is scored as follows: first, the length of the shortest successful trace is compared to the target trace length, TTL. We assume that longer traces, implying more complex chains of action, imply a harder puzzle to solve (since we are agnostic to the mechanic being designed for, this seems a reasonable assumption). Levels whose shortest trace is closer to the target trace length receive a higher fitness. This accounts for half of the total fitness score, normalised between 0 and 0.5, where 0.5 represents a level whose shortest trace is as long as the target trace length. We also compare the number of times the mechanic was used with a target mechanic use variable, TMU, using the logic that frequent use of the mechanic implies a complex solution. This is also normalised between 0 and 0.5. The variables TTL and TTM allow us to adjust both the overall complexity of the level's solution, as well as the ratio of mechanic use to other actions.

Crossover, Mutation, and Evolutionary Parameters. Mutating a level involves the random replacement of abstract level elements with new elements, or the adjustment of parameters such as the length of the lines and box sides, or their starting co-ordinates. We employ uniform crossover by selecting a point in the level grid, and performing crossover of any level elements whose starting co-ordinate lies before that point on the grid. Parameters vary depending on the desired difficulty of the level being designed, which is adjusted by the user through the target trace length and other parameters described in the previous section. A typical run maintains a population of 100 levels, run for 15 generations, with a target mechanic use of 2 and a target trace length of 6. The standard target reaction time is a step of 64ms, with a slower reaction time of 256ms.

4 Results and Evaluation

This section includes three game mechanics, and three levels designed for each, which require the use of the associated mechanic in order to be solved. The player starts in the 'S' position and must reach the exit, marked 'X'. Red tiles kill the player, while black squares are solid ground.

4.1 Mechanic - Gravity Inversion

In Flixel, setting an object's acceleration field is a way to simulate gravity, so a typical platform game normally sets the `acceleration.y` field to a positive integer. Multiplying this value by -1 has the effect of inverting gravity. Figure 2 shows three sample levels.

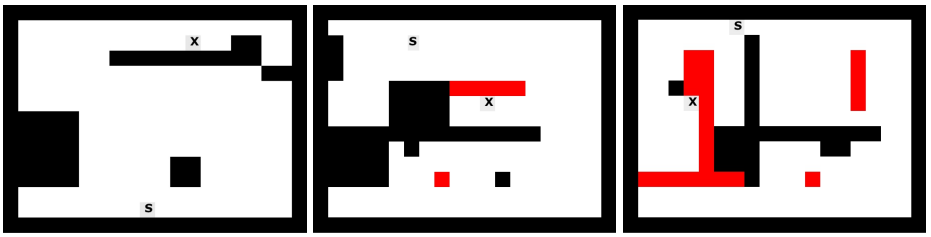


Fig. 2. Gravity inversion mechanic: `INVERTSIGN player.acceleration.y`

4.2 Mechanic - 'Teleportation'

Halving (doubling, if activated again) the player's y co-ordinate allows for teleportation around the game world. This mechanic is unusual, as the classic mental model of a platform game does not match up to the co-ordinate model the game engine uses. Here, (0,0) refers to the top-left corner of the screen. Halving the distance between the player's position and the ceiling has no natural analogue in real-world physics. This makes the mechanic technically usable, but confusing to the player. Figure 3 shows three sample levels.

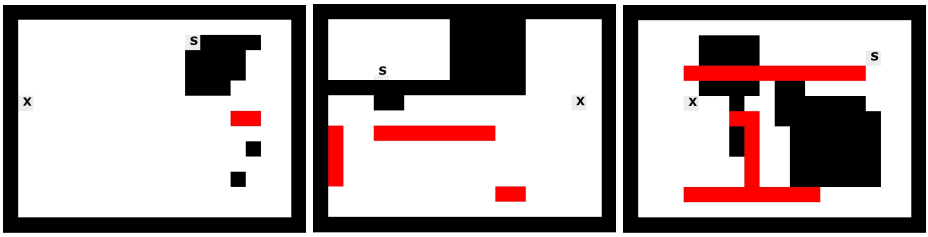


Fig. 3. Teleportation mechanic: `HALVE player.y`

4.3 Mechanic - 'Bounce'

Elasticity affects how an object interacts with the game's physics engine. Elasticity is normally set to 0; any positive number will cause the player to bounce when it collides with things. This mechanic allows the player to become 'bouncy' on demand, gaining momentum to bounce over larger gaps or to greater heights. Figure 4 shows three sample levels.

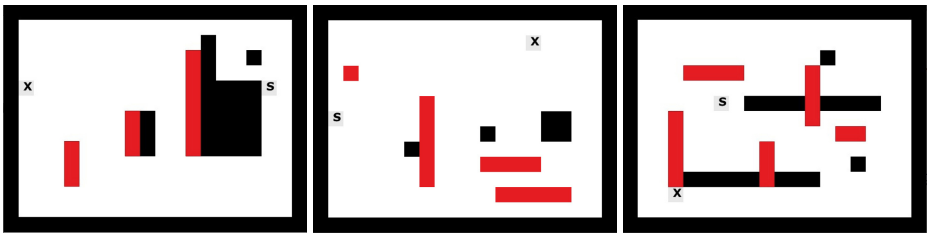


Fig. 4. Bounce mechanic: `ADD 1 player.elasticity;`

4.4 Pilot Study

We performed a small pilot study to gain insight into how some of the mechanics and levels generated would be received by human players. 7 participants with a variety of experience with videogames took part in the study, which consisted of twelve levels split into sets of three sharing a common mechanic. The levels were designed with variable evolutionary parameters; in particular, we attempted to present progressively harder levels to the participants by increasing parameters such as overall solution length and reflex requirements. The mechanics presented to the player were those described above, and a fourth mechanic that allowed the player to halve the effect of gravity. The participants were not told how the mechanics worked, beyond being told they had a special ability they could use.

When asked to describe how they thought the mechanics worked, all five participants failed to describe the teleportation mechanic accurately, and all rated it as unenjoyable. This supports our belief that some mechanics invented by

Mechanic Miner in its current form are confusing for players, because they may refer to elements of the game code that are not in the player's mental model of how the game world works. By contrast, all of the participants rated the gravity inversion mechanic highly, with the hardest level of the set described as "impressive" and "fun" by two of the participants. Comprehension of a mechanic seems understandably integral to a player's enjoyment of using it, and the feedback from the study raises questions as to whether pure utility may be an effective evaluation criteria for a reflection-driven system. Participants were also asked to rank the levels in order of perceived difficulty for each set of three levels. From the responses received, it seems that our metrics (such as the number of times a mechanic must be used in a solution) model difficulty well for some mechanics, but poorly for others - participants tended to agree on which levels were harder than others, but in the case of mechanics such as bouncing, it did not match up to our intended difficulty ordering. This may imply that different mechanics require different metrics, as some may be suited to reaction challenges, while others may be about ordering of actions or puzzle solving.

The study has informed the design of a larger survey that will focus on a subset of mechanics. We hope this will provide more evidence for our hypotheses.

5 Related Work

Our related work covers both ruleset and mechanic invention. A common approach is to select mechanics from an annotated database. The system described in [4] uses a list of known mechanics and game content, associated with words, and uses WordNet to connect input strings to words attached to known game content. This allows for the verb *shoot* to connect to mechanics where the player controls a crosshair shooting at objects, for example. The work in [7] uses smaller mechanical pieces which the authors call *micro-rhetorics* (see [8]). The Game-O-Matic [7] takes sets of these micro-rhetorics and matches them with a human-specified network of entity relationships. In one example in the paper, the relationship *dog needs food* is expressed through two objects labelled 'dog' and 'food', with the player being able to control the dog, tasked with gathering food.

These annotated-database approaches work well where a human designer is able to construct a database of known concepts beforehand, but this limits the autonomy of the system, as well as its ability to surprise us through novel output. This makes it a promising direction for mixed-initiative tasks. However, we are interested in building an autonomous game designer, from a Computational Creativity perspective [3], and this approach is less suited to our interests.

The other main approach to mechanic design is a grammar-based constructive approach that uses templates, into which small sub-mechanic components are inserted to construct high-level rules [6,2]. Here, rule templates describe abstract notions of mechanics (such as an event where two objects collide and effects are applied to them) and then abstract notions of smaller objects/events/effects that are in common videogame vocabulary (game objects such as the player, moving entities, or the game level wall; sub-events such as an object being killed, or

teleported to a new location). This allows for novel composite mechanics to be constructed, but the essential elements of the rules still direct the system towards rediscovering known concepts or slightly elaborating on them. The need for a technique that is not dependent on prior definitions is still evident.

6 Conclusions and Future Work

In this paper we introduced Mechanic Miner, an evolutionary system that generates new game mechanics through Reflection and then validates their usefulness by simulating game playouts. We also showed how the same techniques can be used to design levels that can only be solved using the mechanics invented. Mechanic Miner represents a novel method for both discovering potential game mechanics and exploiting them through level design, without knowing anything about the game engine or the mechanics found during the search. We have found the system to be capable of producing mechanics unexpected to us (such as the mechanic in section 4.3) as well as discovering mechanics similar to those used by human game designers ([11] uses a mechanic similar to 4.1).

The simulation-driven level design system also surprised us by discovering exploits within the game code we had supplied it. The sample game includes a check that the player is touching the floor before allowing them to jump. Mechanic Miner found that by using a teleportation mechanic it could teleport inside a solid wall, which Flixel registers as the same as touching the floor, effectively allowing the player to 'wall jump'. Exploiting game engines in such a way is common in certain game communities, and integral to competitive 'speed runs' of games, where players try to find new ways to shortcut or break the rules of a game. That Mechanic Miner was able to do this, without any anticipation from the system's authors, is an exciting indication that the system may be able to find more complex, emergent game mechanics in future.

Some of the areas suggested for possible future work include:

Game Object Synthesis. Rich game mechanics often affect in-game objects that are created as extensions of the game engine. For instance, some of the mechanics found by Mechanic Miner affect the Player object, which extends Flixel's basic FlxSprite object. By allowing Mechanic Miner to create new types of game object as part of its search process, and then find mechanics that exploit interactions between the new game objects and the game world, we open up more interesting possibilities for discovery.

Richer Reflection and Constraint Generation. Some mechanics only make sense in the context of complementary vulnerability. Mechanic Miner discovered the notion of gravity inversion, which is used in [11]. However, in [11] the player is unable to jump. This simple adjustment allows for many new challenges to be designed. Constraints are just one way that reflection could be leveraged to create more intricate mechanics. Reflection can be used to call methods or run

arbitrary code at runtime, opening up the possibility for more complex, multi-operation mechanics to be constructed beyond field modification.

Emergent Mechanics. In some games, problems are presented to the player that cannot be solved with a single mechanic, but can be completed using a combination two. By modifying Mechanic Miner, we believe it can be used to solve levels using multiple mechanics at a time, potentially discovering scenarios where mechanics can combine to solve new types of problem.

Acknowledgements. Thanks to Julian Benson for discussion of his work on player traces in *Braid*, and Julian Togelius for input into the Mechanic Miner evaluation. Thanks also to the anonymous reviewers for some insightful comments.

References

1. Cardamone, L., Yannakakis, G.N., Togelius, J., Lanzi, P.L.: Evolving Interesting Maps for a First Person Shooter. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A.I., Merelo, J.J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 63–72. Springer, Heidelberg (2011)
2. Cook, M., Colton, S.: Multi-faceted evolution of simple arcade games. In: Proc. of the IEEE Conference on Computational Intelligence and Games (2011)
3. Colton, S., Wiggins, G.: Computational creativity: The final frontier? In: Proc. of the 21st European Conference on Artificial Intelligence (2012)
4. Nelson, M.J., Mateas, M.: Towards Automated Game Design. In: Basili, R., Pazienza, M.T. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 626–637. Springer, Heidelberg (2007)
5. Smith, G., Othenin-Girard, A., Whitehead, J., Wardrip-Fruin, N.: PCG-based game design: creating endless web. In: Proc. of the International Conference on the Foundations of Digital Games, FDG (2012)
6. Togelius, J., Schmidhuber, J.: An experiment in automatic game design. In: Proc. of the IEEE Conference on Computational Intelligence and Games (2008)
7. Treanor, M., Blackford, B., Mateas, M., Bogost, I.: Game-o-matic: Generating videogames that represent ideas. In: Proc. of the Third Workshop on Procedural Content Generation in Games, FDG 2012 (2012)
8. Treanor, M., Schweizer, B., Bogost, I., Mateas, M.: The micro-rhetorics of game-o-matic. In: Proc. of the International Conference on the Foundations of Digital Games, FDG (2012)
9. GNILLEY, Radix (2010), `http://www.gnilley.com/`
10. Offspring Fling, KPULV (2011), `http://offspringfling.com/`
11. VVVVVV, Terry Cavanaugh (2010), `http://www.thelettervsixtim.es`

Generating Artificial Neural Networks for Value Function Approximation in a Domain Requiring a Shifting Strategy

Ransom K. Winder

The MITRE Corporation
rwinder@mitre.org

Abstract. Artificial neural networks have been successfully used as approximating value functions for tasks involving decision making. In domains where a shift in judgment for decisions is necessary as the overall state changes, it is hypothesized that multiple neural networks are likely be beneficial as an approximation of a value function over those that employ a single network. For our experiments, the card game Dominion was chosen as the domain. This work compares neural networks generated by various machine learning methods successfully applied to other games (such as in TD-Gammon) to a genetic algorithm method for generating two neural networks for different phases of the game along with evolving a transition point. The results demonstrate a greater success ratio with the method hypothesized. This suggests future work examining more complex multiple neural network configurations could apply to this game domain as well as being applicable to other problems.

1 Introduction

Artificial neural networks have proven to be a robust computational method for data modeling and pattern recognition. An application where this method has enjoyed successful implementation is in decision making. While several successes have been achieved using single neural networks as approximators of nonlinear value functions, a question arises as to whether an individual, traditional feed-forward neural network is sufficient for handling an environment where there are shifts in strategy required as the overall state changes. In this situation, it is hypothesized that multiple neural networks that approximate different policies and are gated by simple conditions will be more competitive. Given the difficulty of learning such a system, evolutionary computation (specifically genetic algorithms) is an attractive method for discovering neural networks and state transitions where the neural network applied should change.

Neural networks as decision makers have been applied impressively for certain games. This paper intends to apply the genetic algorithm method described to a deck-building card game called Dominion, where a change in policy during play makes for an effective strategy. The goal is to generate a neural network to accurately approximate the value of game states in order to make decisions. These neural networks will take as input possible and predicted game states and output value judgments on those

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 294–303, 2013.
© Springer-Verlag Berlin Heidelberg 2013

states to make a decision. Ideally, when used throughout a game, these networks provide an optimal strategy. This method will be compared against reinforcement learning and hill-climbing techniques, which have been demonstrated successfully in other domains.

2 Background

2.1 TD-Gammon and Variants

TD-Gammon [15-16] still remains a marked success in the application of machine learning to the problem of developing automated systems to play games. The original application of TD-Gammon demonstrated reinforcement learning [14] in the game of backgammon, using a neural network trained with backpropagation as an approximation of a nonlinear function of a game state's value. Reinforcement learning has been shown to achieve measures of success in other game domains such as simpler, solved games like Tic-Tac-Toe and Connect 4 [6] and more complex ones such as The Settlers of Catan [11], but the process is often brittle with respect to the feature space used, requiring this to be finely tuned [8]. The method has been successfully enhanced with genetic algorithms for game environments such as Pac-Man [5].

After TD-Gammon, other machine learning methods have been applied against backgammon. [12] found a simple hill-climbing method used in conjunction with player co-evolution based on relative player fitness was also capable of achieving a measure of success at learning the game. This work suggested it was specifically self-play over iterations of games that drove the success in learning. Yet this approach was still found to be subject to being trapped in local optima as it searched the fitness space of neural network policies [10]. Another learning strategy capable of a wider search of the fitness space that has been demonstrated successfully against backgammon is genetic programming [1], again involving play between learning players.

The implication here is that employing evolution and self-competition is often a part of successful methods. Evolutionary computation methods such as genetic programming or genetic algorithms have enjoyed success across many game-related scenarios, including completely player deterministic games such as chess and chess end-games [4,7], checkers [3], and Go [2]. There is also recent evolutionary computation work applied to Dominion, evolving card sets to balance the game for players of different skill levels [9].

2.2 Domain Space – Dominion

Dominion [17-20] is a card game designed by Donald X. Vaccarino and published by Rio Grande Games. In the game, players start with an initial seed deck of low-powered cards (3 Estates and 7 Coppers) and grow their deck by adding cards to it from a common Supply with the goal of achieving the highest number of points by the game's end. Important card types in this Supply include *treasure*, the currency by which players buy new cards (i.e., Copper, Silver, and Gold with cost/values of 0/1, 3/2, and 6/3, respectively), *victory*, cards that are only worth points at the end of the

game (i.e., Estates, Duchies, and Provinces with cost/values of 2/1, 5/3, 8/6), and *action*, cards which allow players to take actions (in our scenario: Adventurer, Bazaar, Chapel, Conspirator, Festival, Moneylender, Village, Woodcutter, Worker's Village, and Workshop with rules available in [17-20]). These action cards are a subset of many possible *kingdom* cards that can be included in varied Dominion games, but those not listed do not apply here.

Dominion exhibits an apparent difference between early strategies and late-game strategies that can be expressed simply. Only victory cards are worth points at the end of the game, but they take up space in a player's deck and hand during the game, diminishing a player's efficiency. It is not trivial to determine what cards to acquire and when to acquire them to ensure a higher score than an opponent at the end of the game. This indicates the early strategy should be to increase the player's power to buy points at the end by gaining the most effective treasure and kingdom cards in a good distribution. Capturing this change in strategy is important in a successful neural network for assessing a state's value when applied to games of Dominion.

3 Methods

The feed-forward neural network is applied to the game state by a player whenever the player is required to make a choice. Choices include taking an action, making a decision implied by an action taken, and buying a card to add to the deck. The input values to this network are the counts of the different cards in varied game conditions. These conditions include the counts of cards in the Supply, in the player's hand, in the player's discard pile, in the player's draw pile, in the player's play space (that is, cards the player has played down this turn), or in the opponent's deck. Further, there are input values representing the number of actions a player has, the number of buys the player can perform, and the amount of treasure available for those buys. Finally, there is also an input for a count of unknown cards, as some actions require the player to draw an unknown card from the deck. These inputs, which represent a game state, are fully connected to a hidden layer of 200 nodes. These in turn are fully connected to an output layer. A sigmoid function is applied to the sum of nodes and weights for all incoming weights to a node to ensure a result falls between 0 and 1.

Players have two consecutive phases, A and B, where actions are taken and cards are purchased, respectively. During a player's turn where an action can be taken (phase A), the players apply all predicted states from legal actions against the neural network. The resulting outputs from the network are then compared. The highest value is considered to be the optimal predicted state and the player will take that action. A state where no action is taken is also evaluated. If this state is optimal or a player has no further legal actions, then the player applies the same network to all predicted states resulting from buying any cards that can be afforded (phase B). If a state where nothing is bought is judged optimal or there is insufficient currency to buy any cards or the player has no buys left, then the player will discard the hand and play space and draw a new hand for next turn.

Although the common Supply and the knowledge of what players have added to their deck is open to all players, much of the information of the game state is hidden from a player at any given point in the game. Decks are shuffled into an unknown draw-order and the content of an opponent's hand is not known. This makes the predictability of the game highly limited as the possibility of what the opponent is capable of doing next turn—let alone what a player himself is able to do next turn—is highly dependent on the random card draw. While a deck's overall contents are known and probabilities of card draws can be ascertained, the combinations of cards are often more important than the likelihood of any one card emerging, and this is more difficult to predict. Turns are independent of actions taken immediately preceding them and, therefore, prediction past this turn is unlikely to be helpful.

The reinforcement learning and backpropagation used here are similar to what was applied in TD-Gammon. Offline temporal-difference learning is used for reinforcement learning here. Experiments where only the final state is rewarded or punished are also attempted. Resilient backpropagation, or RPROP [13] is applied here with η- and η+ values of 0.5 and 1.2, respectively. Games are played with a variety of different values for the key parameters, including the learning rate for the backpropagation, the temporal difference learning rate, and the type of opponent played against, which can be a co-learning player, a mutant of the player (generated by Gaussian noise applied to its neural network's weights and normalized by the overall network size), or the player itself.

A simpler method for learning that involves simply hill-climbing to a state that has been exhibited in play to be more successful appears to be a reasonable alternative for consideration. Again in these experiments, the player plays against a mutant of itself generated in the same manner as described above, and if the mutant player is more successful, the player's neural network weights are then adjusted in the direction of the mutant by a learning rate (default value of 0.1).

Genetic algorithms as applied to this problem bear some resemblance to the hill-climbing effort. This is performed with an initial population of players with random neural networks. Each player is pitted against the rest of the population in pairwise games. The population of winners is taken to be the subset of these players with the highest rate of victory. These winners are kept for the next generation and also used to generate a new population of mutants and partial mutants, where there is some probability that all network weights will be adjusted by Gaussian noise normalized to the network size (mutant) and a separate probability that any individual weight will be adjusted in the same manner (partial mutant). By the end of the evolutionary process, the final best player is taken to be the equivalent of the player produced by the other methods and is used for a final assessment of the methods' efficacy.

This process can be enriched by allowing different subpopulations to compete in tournaments but have their own pool of winners, which can help to mitigate a problem that occurs where a single winner and its offspring are powerful enough to become the entire population, making the population too similar after several generations.

It seems likely in Dominion that there is a difference between early strategies and late-game strategies. Because victory cards are only valuable after the game is over, and they are the only thing that are valuable at the end, a simple assessment of the

game indicates that the early strategy should be to increase the player's power to buy points at the end, by gaining treasure and effective kingdom cards. A single nonlinear function approximation, such as the neural network represents, might have difficulty representing this change in policy. Thus, experiments are performed that attempt to evolve two feedforward neural networks as well as the threshold (in terms of the turn number) at which the policy changes from one to the other. It is expected that this might be more effective at capturing a change in strategy.

In order to gauge the success of a strategy learned by the neural network, it is necessary to establish baseline policies for taking actions and buying cards in the game. Ideally, the neural networks generated should be able to achieve success against these baselines in a significant percentage of games played. These experiments compare against three different baselines of varying quality or depth, including a policy that builds a deck randomly and takes random actions (Random), a policy that builds a deck with only treasure cards and victory cards (Money), and a policy that builds a deck with only treasure cards, victory cards, and one Chapel card (Chapel).

The Random policy takes random actions from any available actions in the hand (taking no action being viable) and then buys a random card that the current hand can afford (buying no card being viable). This policy is highly naïve and expected to fail, considering the space of available cards is wide and that, without direction, the score is apt to stay stagnant or fall if cards of negative value are purchased.

The other policies are less naïve and based on known successful simple strategies in Dominion, where an emphasis is placed on buying only treasure cards as opposed to relying on action cards and then changing the emphasis to purchasing high-valued victory points. Because one of the ending conditions is when the highest valued cards are all purchased, these strategies are designed simply to pursue building a deck that will purchase those cards as quickly as possible.

The Money policy only purchases the most expensive affordable treasure card on each turn until two Gold cards are present in the deck. Once that threshold is reached, the policy changes to prioritize purchases such that the most expensive treasure or victory card is bought each turn. The intent here is to avoid purchasing victory cards, which do not help in buying cards until later, to ensure that there are a sufficient number of treasure cards in the deck to buy higher cost cards. The threshold of two Gold cards in the deck is an estimate of this point.

The Chapel policy will purchase a Silver card and a Chapel card on the first two turns. The Chapel card is an action that can permanently remove up to four cards in a player's hand from the deck. Subsequent turns will follow the policy of purchasing Provinces, Gold, and Silver cards in that order of preference given what a hand can afford. After 25 turns, the policy will prefer to buy Duchy cards over Silver cards. The intent here is similar to the above policy, but with the added power of the Chapel. Hands that contain the Chapel will remove any Estate cards or Copper cards, unless the Copper cards are combined with higher valued treasure cards to ensure the purchase of a Province, Gold, or Silver. This strategy attempts to remove low-valued cards from the deck completely, increasing the efficiency of the average hand, which will have fewer Estate or Copper cards that provide little value relative to the others over the course of the game.

Table 1 shows the mean results of example games of these policies playing each other, including the percentage of games won and the average number of total hands of any player in the game. As anticipated, the Random policy is slower and achieves a far lower score than the other two. Because cards with a negative score have no cost, the overall score of a Random policy deck often drops below zero. Of the scripted Money and Chapel policies, the Chapel policy is the faster and more apt to win.

Table 1. Baseline policies compared against one another averaged over 10000 games for each pairing of Player 1 (P1) and Player 2 (P2)

P1 Policy	P2 Policy	P1 Win%	P2 Win%	P1 Score	P2 Score	# Turns
Random	Random	49.6	50.4	2.4	2.4	70.7
Random	Money	0.0	100.0	-0.2	60.0	57.6
Random	Chapel	0.0	100.0	-0.2	65.9	65.0
Money	Money	50.2	49.8	32.0	31.8	36.7
Money	Chapel	35.7	64.3	28.8	32.0	34.3
Chapel	Chapel	50.3	49.8	27.8	27.8	31.9

These policies are not intended to be finely tuned, and further alterations that would enhance their performance could be made to their algorithms. They are intended to serve as baselines against which learned policies can be benchmarked.

4 Results and Discussion

Several variations on each of the described learning methods were applied. For brevity's sake the *best* results achieved with each method assessed against the baseline strategies are listed in Table 2, with the exception of reinforcement learning, where the best of multiple variations are shown as the changes make significant differences in how learning proceeds.

Player R/FS was trained with reinforcement learning using a backpropagation learning rate of 0.01 and only learned on the final state of the game, playing its games against a mutant of itself. The mutant was generated from the current neural network of the player before each learning iteration. Player R/TD similarly used backpropagation and competition against a mutant, but also applied temporal difference learning with a decay rate of 0.9999 to diminish the reward as states went from the final state back to the beginning. Player HC was trained using the hill-climbing method with a rate of change toward a winning mutant of 0.1. Player GA/S was trained in a genetic algorithm tournament where there were 5 separate subpopulations of 10 that were pitted against each other with a single winner per population after each tournament that was used to seed the following generation. The chance of complete mutation was 0.1 while the chance of mutation in a descendant for any given network connection was 0.01. Player GA/M was trained using the genetic algorithm method that evolved two feedforward neural networks and the turn threshold where the network utilized changed from the first to the second. In these results there were 100 members of a population with the 50 highest scorers carried forward to the next generation. The rates of mutation here were the same as for player GA/S.

Table 2. Performance (averaged over 1000 games) of players and their neural network policies against the baselines as well as between one another for the higher scoring policies

P1 Method	P2 Policy	P1 Win%	P2 Win%	P1 Score	P2 Score	# Turns
R/FS	Random	100.0	0.0	51.0	-0.1	51.6
R/FS	Money	39.0	61.0	28.6	30.0	35.3
R/FS	Chapel	29.7	70.4	25.8	29.9	33.2
R/TD	Random	100.0	0.0	11.0	-2.4	95.8
R/TD	Money	0.0	100.0	11.0	59.4	57.0
R/TD	Chapel	0.0	100.0	11.0	65.8	65.1
HC	Random	100.0	0.0	63.4	-0.2	68.2
HC	Money	70.3	29.7	37.3	33.5	37.9
HC	Chapel	44.8	55.2	32.6	33.3	35.5
GA/S	Random	100.0	0.0	51.6	-0.1	51.7
GA/S	Money	47.9	52.1	29.6	29.4	34.9
GA/S	Chapel	38.5	61.6	26.4	29.3	32.5
GA/M	Random	100.0	0.0	63.0	-2.2	66.2
GA/M	Money	76.1	23.9	38.0	32.7	37.2
GA/M	Chapel	61.2	38.8	33.7	31.8	34.6

Table 3. Distribution of cards in players' decks averaged over 1000 games against the Chapel strategy. Cards in the Supply that were never purchased are not listed.

	R/FS	R/TD	HC	GA/S	GA/M
Copper	7.000	13.765	7.024	4.952	7.000
Silver	7.200	17.614	5.675	4.060	3.874
Gold	5.249	--	3.635	4.653	3.423
Estate	3.000	11.000	3.519	3.411	5.114
Duchy	--	--	2.959	--	2.524
Province	3.805	--	3.365	3.834	3.507
Adventurer	--	--	0.020	--	--
Bazaar	--	--	--	2.716	--
Conspirator	--	--	0.005	--	0.081
Festival	--	--	--	--	2.342
Moneylender	--	--	--	0.550	--
Woodcutter	--	--	1.185	--	0.001
Worker's Village	--	--	0.007	--	--
Workshop	--	--	--	--	0.039

Table 3 further lists the distribution of cards witnessed in the players' decks averaged over the 1000 games played against the Chapel policy. Some cards appear in negligible quantities (e.g. Adventurer, Workshop), but others are consistent purchases that distinguish the strategies from one another and are revealing about how the neural

networks drive the player. In most cases a modest number of kingdom cards are added to a deck, outweighed by the total of treasure and victory cards purchased.

While each of the depicted players is able to beat the trivial baseline of the Random strategy, none is able to consistently best the Money or Chapel strategies (Table 2). Player R/TD's inability to win any games against these strategies suggests that temporal difference learning is not suited for this game domain. Temporal difference methods that pitted a learning player either against a co-learning player or an opponent with an identical neural network resulted in performances that were not even capable of defeating the Random strategy consistently. Player R/FS's inability to win more than 50% of the games against the Money or Chapel strategies seems to further indicate that backpropagation applied against trying to learn a policy based on a final state is a flawed method for Dominion.

The other strategies find more success, though neither the J nor the GA/S player are able to win more than 50% against the more efficient Chapel strategy. Only player GA/M, who has evolved two neural networks to approximate an early and a late game policy, is able to win well over 50% of the games against both the Money and Chapel strategies. This suggests that there is indeed a need for a change in policy as the game progresses, which is difficult to capture in a single neural network approximation.

It is worth noting that there is some variance in the expense required to find the neural networks. Generating the genetic algorithm solutions takes considerably longer than the other methods because tournaments must be run between many players to discover the best neural networks. However, granted that the most successful solution falls into this category, the expense is justified.

What is intriguing about the most successful method (GA/M) is that fewer treasure cards are purchased here than with the other methods, and a wider variety of victory cards are purchased (Table 3). A diversity of action cards are purchased, but only one (Festival) is purchased with any consistency. Therefore rather than an optimized version of the Money or Chapel policies, a different and better strategy appears to have been uncovered by the genetic algorithm method, which is encouraging when considering applying this method to discover strategies on other sets of 10 kingdom cards.

Notable for its absence is the purchase of the Chapel card by any of the best observed strategies. Rather than suggesting the Chapel is not part of an optimal strategy, it is likely that these methods have difficulty in discovering its complex use. The Chapel has many possible states must be assessed when performing the prediction because of the varied combinations of cards the Chapel can remove.

5 Conclusions

This paper has examined an array of machine learning methods applied against the Dominion card game, demonstrating methods that have been successfully applied to backgammon and other games in the past (reinforcement learning, hill-climbing) to be less successful in this domain, at least with the features selected. One expected limitation of these methods was due to a suspicion that a single feed-forward neural network is unable to capture a key change in strategy that is often effective when playing

a game of Dominion, namely that a player should at some point switch from building a deck with effective actions and high-valued treasure cards to buying victory cards. Applying a genetic algorithm method that evolved two neural networks and the time threshold at which a change in policy occurs was able to discover policies that could effectively defeat the human-defined baselines that the other methods could not beat more than 50% of the time. This is an encouraging result and potentially could be applied to other scenarios, games or otherwise, where a change in policy is required, either at a certain point in time or when certain criteria are met.

Looking forward, there are many avenues of future work both in this game domain and using the methods applied here as a basis. First and foremost, the evolution of multiple neural networks to act as approximators of nonlinear value functions and the decision points where switching engines occurs can be enriched and applied to other domains. Allowing for evolution to allow also for a variable number of networks and switching points would be the first step to subsequently evolving modular neural networks that have more flexibility in their construction and flow of information depending on the current state in which the neural network is operating and the predicted states. It can also lead to evolving networks that, instead of providing judgments on a state's quality, act as gating mechanisms to those neural networks that evaluate the quality of states. This applies across Dominion and other domains. Further, while the change in neural network policies achieved here occurs based on a turn threshold, in other scenarios this can be evolved to be driven by certain intermediate states arising or conditions being met. Discovering those states and conditions can be evolved.

While the best examples of the genetic algorithm found strategies that exceeded the baselines' win rates, it is still the case that some of these were only able to achieve lesser success, ending the evolutionary process with a different strategy. This indicates that there are pitfalls in the way of local maxima that strategies can enter which can be difficult to escape, even for the genetic algorithm method. While repeated evolution from different initial states allows for further exploration of the space of neural networks, future work will explore other methods of escaping local maxima.

While it is encouraging to learn approximate policies can be induced using these methods on neural networks for an individual game setup as demonstrated here, there are presently approximately 14 quadrillion setups of Dominion depending on which ten kingdom cards are included from over the 190 possible. What has been learned here will not extend to games where the chosen kingdom cards are not part of the Supply. Discovering a general policy for Dominion with genetic algorithms (effectively creating a GA-Dominion) is an intriguing prospect, but requires a greater depth of input feature representation and state interpretation than is achieved here.

References

1. Azaria, Y., Sipper, M.: GP-Gammon: Using Genetic Programming to Evolve Backgammon Players. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 132–142. Springer, Heidelberg (2005)
2. Cai, X., Wensch, D.: Computer Go: a Grand Challenge to AI. SCI, vol. 63, pp. 443–465. Springer, Heidelberg (2007)

3. Chellapilla, K., Fogel, D.: Evolving Neural Networks to Play Checkers without Relying on Expert Knowledge. IEEE Trans. on Neural Networks 10(6), 1382–1391 (1999)
4. Fogel, D., Hays, T., Hahn, S., Quon, J.: A Self-Learning Evolutionary Chess Program. Proc. of the IEEE 92(12), 1947–1954 (2004)
5. Galway, L., Charles, D., Black, M.: Improving Temporal Difference Game Agent Control Using a Dynamic Exploration Rate During Control Learning. In: Proc. of IEEE Symposium on Computational Intelligence and Games, pp. 38–45 (2009)
6. Ghory, I.: Reinforcement Learning in Board Games. Technical Report CSTR-04-004. Department of Computer Science, University of Bristol, UK (2004)
7. Hauptman, A., Sipper, M.: GP-EndChess: Using Genetic Programming to Evolve Chess Endgame Players. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 120–131. Springer, Heidelberg (2005)
8. Konen, W., Bartz-Beielstein, T.: Reinforcement Learning for Games: Failures and Successes. In: Proc. of GECCO 2009, pp. 2641–2648 (2007)
9. Mahlmann, T., Togelius, J., Yannakakis, G.: Evolving Card Sets Toward Balancing Dominion. In: IEEE World Congress on Computational Intelligence (2012)
10. Oon, W., Henz, M.: M2ICAL Analyses HC-Gammon. In: ICTAI 2007, pp. 28–35 (2007)
11. Pfeiffer, M.: Reinforcement Learning of Strategies for Settlers of Catan. In: International Conference on Computer Games: Artificial Intelligence, Design and Education (2004)
12. Pollack, J., Blair, A.: Co-evolution in the Successful Learning of Backgammon Strategy. Machine Learning 32(3) (1998)
13. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP algorithm. In: IEEE Conference on Neural Networks, pp. 586–591 (1993)
14. Sutton, R., Barto, A.: Reinforcement Learning. MIT Press, Cambridge (1998)
15. Tesauro, G.: Practical Issues in Temporal Difference Learning. Machine Learning 8, 257–277 (1992)
16. Tesauro, G.: Temporal Difference Learning and TD-Gammon. Comm. of the ACM 38(3) (1995)
17. Vaccarino, D.X.: Dominion, Game Rules, `http://www.riograndegames.com/uploads/Game/Game_278_gameRules.pdf` (retrieved October 2012)
18. Vaccarino, D.X.: Dominion: Intrigue, Game Rules., `http://www.riograndegames.com/uploads/Game/Game_306_gameRules.pdf` (retrieved October 2012)
19. Vaccarino, D.X.: Dominion: Seaside, Game Rules, `http://www.riograndegames.com/uploads/Game/Game_326_gameRules.pdf` (retrieved October 2012)
20. Vaccarino, D.X.: Dominion: Prosperity, Game Rules, `http://www.riograndegames.com/uploads/Game/Game_361_gameRules.pdf` (retrieved October 2012)

Comparing Evolutionary Algorithms to Solve the Game of MasterMind

Javier Maestro-Montojo[1], Juan Julián Merelo[2], and Sancho Salcedo-Sanz[2,⋆]

[1] Department of Signal Processing and Communications, Universidad de Alcalá, 28871 Alcalá de Henares, Madrid, Spain
sancho.salcedo@uah.es
[2] Departamento de Arquitectura y Tecnología de Computadores, Universidad de Granada, Granada, Spain
jmerelo@geneura.ugr.es

Abstract. In this paper we propose a novel evolutionary approach to solve the Mastermind game, and compare the results obtained with that of existing algorithms. The new evolutionary approach consists of a hierarchical one involving two different evolutionary algorithms, one for searching the set of eligible codes, and the second one to choose the best code to be played at a given stage of the game. The comparison with existing algorithms provides interesting conclusions regarding the performance of the algorithms and how to improve it in the future. However, it is clear that Entropy is a better scoring strategy than Most Parts, at least for these sizes, being able to obtain better results, independently of the evolutionary algorithm.

1 Introduction and State of the Art

Mastermind is a classic board game for two players, invented by M. Meirowitz in 1970. One of the players acts as a *codemaker*, and the other one as *codebreaker*. The codemaker starts the game by choosing a secret code, i.e. a sequence of P different pegs of colors, out of N possible colors (repetitions are allowed). The codebreaker tries to guess the code by means of a set of guesses. After each guess, the codemaker gives an answer to the guess, in terms of two numbers: a number of black pegs for those elements in the guess that coincide in color and position with the ones in the secret code, and a number of white pegs for those elements in the guess that coincide in color, but not in position with the ones of the secret code. Figure 1 shows an example of the game, with the secret code and two guesses (with their respective answers).

Mastermind is a challenging problem from the algorithmic point of view. The game caught the attention of many researchers by the end of the seventies so there are a good amount of works dealing with this problem in the literature. Different techniques have been applied to the solution of the problem: full enumeration approaches, heuristics based on simple rules and also meta-heuristic algorithms.

⋆ Corresponding author.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 304–313, 2013.
© Springer-Verlag Berlin Heidelberg 2013

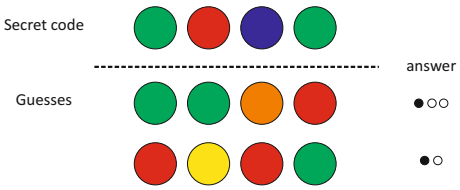


Fig. 1. Example of a Mastermind game

Among full enumeration techniques, the pioneering proposal by Knuth et al. [1] (known as Worst Case strategy) was able to succeed for $N = 4, P = 6$ in less than five moves, and was based on choosing the combination that minimizes the maximum number of remaining eligible codes. Other approaches to Mastermind applying full enumeration techniques are [2,3]. In [4], a full enumeration technique was mixed with Information Theory in order to obtain the best enumeration strategy. This approach was named as "Max Entropy" and is currently in use in different algorithms for the Mastermind. In [5] the strategy known as "Most Parts" were introduced and applied to obtain a solid exhaustive algorithm to the problem. More recently, in [6] and [7], a variant of full enumeration via depth-first search is proposed, and in [8] different new exhaustive search strategies are proposed, after a full statistical revision of the problem and alternative exhaustive approaches.

In a different approach to the problem, there are different heuristic-based approaches which try to obtain good solutions to the problem with a short computation time. The first approach of this kind is [9], where the codebreaker constructs a list of colors elements, and for each guess, he starts from the last examined position in the previous guess until find an eligible code. The approach in [10] is similar, with a variant in the ordering colors in the list. In [11] a hill climbing approach is proposed, which minimizes the average number of guesses and also the number of evaluated codes. The most recent approaches to Mastermind relay on meta-heuristics algorithms. The first evolutionary-based algorithms for tackling the Mastermind game appeared in the last 90's and first years of 2000 [12,13,14]. More recently, in [15] a stepwise evolutionary approach was first proposed. That approach played any eligible code as soon as it was found by the algorithm. A subsequent approach also based on evolutionary algorithm was presented in [17], where an evolutionary algorithm selects a number of eligible codes, and then the one played is selected by a mechanism of Expected Size. The most recent approach to Mastermind with evolutionary computation was presented in [16], where new strategies to improve the performance of evolutionary algorithms in Mastermind were suggested.

In this paper we propose a novel evolutionary approach to the Mastermind based on a nested hierarchical evolutionary search. The proposed approach solves the problem by an initial evolutionary search of the eligible set, and a second evolutionary approach to obtain the best possible guess to play, based on different predictive strategies. We present a comparison study of the performance of the

proposed nested hierarchical approach with the algorithm in [18], in which we introduce a new variant that uses a different scoring algorithm.

The structure of the rest of the paper is the following: next section presents some important notation and definitions used in evolutionary algorithm proposed. Section 3 presents the nested hierarchical evolutionary algorithm introduced in this paper, and also reviews the main of the stepwise approach in [16]. Section 4 shows the results obtained by both approaches in a number of numerical experiments, to show the performance of both algorithms. Section 5 closes the paper with some final remarks and conclusions.

2 Mastermind: Basic Notation and Definitions

The usual notation for a Mastermind game is $G(P, N)$, where P stands for the number of pegs that the secret code must have, and N the number of different colors that can be chosen to form the code. A classical encoding for a given guess ($\mathbf{g}$) consists of a string of P numbers, $P \in \{1, N\}$, representing the colors of the secret code. In addition, we denote by X the number of black pegs in the codemaker answer, and Y the number of white pegs. Note that if the code is finally guessed, then $X = P$ and $Y = 0$.

Definition: We denote the *eligible set* after n guesses (E_n) as the subset of global combinations that can still be the secret code.

As an example, Figure 2 shows the way in which the eligible set changes after different guesses in a game. Note that $E_0 \supset E_1 \supset E_2 \supset E_3 = \{(3, 2)\}$. Finally the only possible combination that can be the secret code is $(3, 2)$. Note also that a given guess g_{n+1} (after the current one $\mathbf{g}_n$), will divide E_n in different subsets, depending on the answers (X and Y) that can be given to the try. Figure 3 shows this process. We denote the partitions produced by a try as E_{n+1,r_i}.

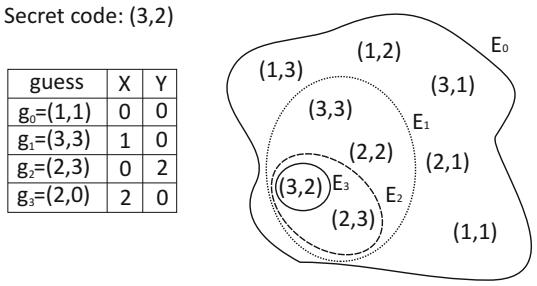


Fig. 2. Example of the eligible set changes after different guesses in the game

Definition: We denote matrix XY for a given guess $\mathbf{g}_{n+1}$ as the matrix formed by all the cardinals of the eligible subsets partitions E_{n+1,r_i}, i.e.,

$$XY = \{|E_{n+1,r_1}|, |E_{n+1,r_2}|, \ldots, |E_{n+1,r_k}|\},$$

where k is the number of possible answers to $\mathbf{g}_{n+1}$.

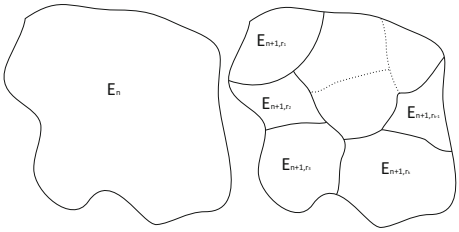


Fig. 3. Example of eligible subsets partitions once a given guess has been played

3 Compared Algorithms

In this section we will present the two algorithms that are being compared in this paper, BS in Subsection 3.1 and Evo++ in Subsection 3.2

3.1 Proposed Nested Hierarchical Evolutionary Approach

The proposed nested hierarchical evolutionary approach is based on a significant modification of the evolutionary approach in [17]. In this case, the selection of the combination played (guess $\mathbf{g_{n+1}}$) is obtained by a second evolutionary algorithm, launched for the eligible set E_n from the first evolutionary approach. The final algorithm is therefore a combination of two (nested and hierarchical) evolutionary algorithms, the first one (Search EA) focused on obtaining the best possible eligible set, and the second one (Decision EA) on providing the optimal guess $\mathbf{g_{n+1}}$ among the elements in E_n. Both algorithms must be run several times to obtain a final solution to the Mastermind. Note that the best individual in the Decision EA provides the information necessary to calculate the fitness for another run of the Search EA.

The Search EA can be described as follows: For a generic Mastermind game $G(P, N)$, and a first played guess g_1, a first initial population is randomly generated containing the different tempted solutions (codes) to the problem. The Search EA aims to obtain a set of codes belonging to the eligible set E_n, taking into account the sequence of n previously played codes ($n = 1$ in the first

generation of the algorithm). The fitness value of each code is calculated by using the expression proposed in [17]:

$$f(\mathbf{C}) = \sum_{i=1}^{n} 2 \cdot P - |X_i' - X_i| - |Y_i' - Y_i|, \tag{1}$$

where $\mathbf{C}$ stands for the code to be evaluated, n is the number of codes previously evaluated, X, Y are the responses of the code evaluated with respect to the real secret code, and X', Y' stand for the responses give if we consider C as the secret code. Note that when X_i' and Y_i' are equal to X_i and Y_i, the value of $f(\mathbf{C}) = 2 \cdot P \cdot n$, and C belongs to the eligible set E_n.

The evolutionary operators applied in this Search EA are one and two points crossover (selected with a probability of 0.5 at the time of their application), and direct random, swapping and inversion mutations (these applied with a small probability). In addition, an operator of hyper-mutation, similar to the one introduced in [15] is considered after a number of generations. Hyper-mutation consists of re-starting a high percentage of the population to random individuals if some conditions are fulfilled. The selection procedure to choose the parents is the roulette wheel, and a mechanism of elitist population replacement between the initial and the new generated one (after the application of the evolutionary operators) is considered. To do this, the initial and new population are merged together, a percentage of the best individuals are kept, and the rest of the population is completed by randomly choosing the remaining individuals. A diversity mechanism is also applied, in such a way that if two elements in the population are equal, one of them is removed and replaced by a randomly generated individual.

Note that we have set upper and lower bounds for the eligible set size. At the beginning of the algorithm, this size can be huge, whereas in the final steps of the algorithm the eligible set size will be much smaller. In this latter case, the Search EA is stopped when a maximum number of generations have been reached, if there is at least one element in the eligible set.

Once the Search EA has finished its run n, an eligible set outcome E_n is obtained. The Decision EA is launched at this stage, in order to come up with the best guess g_{n+1}, out of the elements of E_n. This process uses specific strategies applied over the eligible set E_n in order to estimate the predictive power of each code in E_n. These strategies are the one-step anticipation strategies, i.e. *Expected Size* (ES), *Most-Parts* (MP), *Worst Case* (WC) and *Entropy* (EN). The Decision EA uses the same encoding that the Search EA, so we have considered similar operators in both algorithms. The fitness function used depends on the anticipation strategy considered. Note that anticipation strategies determine the prediction power for a combination by means of operations in the XY matrix of the game. Recall that, for a specific point of a game $G(P, N)$, XY represents the way in which a given guess divides the eligible set E_n. Let M be the XY

matrix of an specific guess g, the fitness functions defined for each anticipation strategy are the following:

$$f(M)_{ES} = \sum_{i=1}^{k} \frac{|E_n|}{|E_{n+1,r_i}|}, \tag{2}$$

$$f(M)_{MP} = \sum_{i=1}^{k} 1, \tag{3}$$

$$f(M)_{EN} = \sum_{i=1}^{k} -\frac{|E_{n+1,r_i}|}{|E_n|} \cdot \log\left(\frac{|E_{n+1,r_i}|}{|E_n|}\right), \tag{4}$$

$$f(M)_{WP} = \frac{|E_n|}{\max(|E_{n+1,r_1}|, |E_{n+1,r_2}|, \ldots, |E_{n+1,r_k}|)}, \tag{5}$$

In addition, note that $f(M) = 0$ if $|E_n| = |E_{n+1,r_j}|$ for any $j \in 1, \ldots, k$, and $|E_n| \neq 1$ (those combinations that do not reduce the eligible set size do not have any predictive power).

3.2 Stepwise Evolutionary Approach for Comparison

This approach has been already presented in [16]. It uses a single evolutionary algorithm to first find feasible solutions and then continue evolution for a few generations within the set of feasible solutions so that there is a certain degree of optimization of the score of feasible solutions. In that sense, it is also similar to Berghmańs although the way of obtaining the score is completely different.

The fitness function used by this method combines the two evolutionary algorithms mentioned above in a single one. First factor is the *distance to feasibility* used by the Search EA above. Instead of Equation (1), this formula is used:

$$f(\mathbf{C}) = \sum_{i=1}^{n} |X'_i - X_i| - |Y'_i - Y_i|, \tag{6}$$

Unlike above, C is eligible when f($\mathbf{C}$) is 0. In that case, the second factor is considered. This second factor includes the feasibility score, but instead of using the expected value as in the Decision EA above (expressed in Equation (2) and subsequent), two different scoring methods are used: Entropy (called EN above) and Most Parts (MP). These two factors (which we can call "search" and "decision" terms, f_S and f_D are then combined in a single fitness function as follows:

$$f(\mathbf{C}) = f_D(C) + 1 - \min_k\{f_S(k)\}$$

where k runs over all non-feasible combinations in the population. This implies that the worst non-feasible combination will have fitness equal to 0, and all feasible combinations have a fitness that is above all non-feasible combinations.

Table 1. Values for the Evo++ parameters that obtain the best result. Permutation, crossover and mutation are *priorities*; they are normalized to one to convert them to application rates. In practice, crossover will be applied to 80% and mutation and permutation to 10% of the newly generated combinations each.

Parameter	Value
Crossover	8
Mutation	1
Permutation	1
Replacement rate	0.75
Tournament size	7

Table 2. Values for the BS parameters used in the simulations

Parameter	Value
One point Crossover Prob.	0.5
Two points Crossover Prob.	0.5
Mutation	0.03
Inversion	0.02
Swapping	0.03

This evolutionary algorithm has been also optimized so that the maximum number of evaluations achieved in the worst case is minimized. The only change with respect to results already published is that, besides using the MP strategy, we have used here for the first time EN. Results will be shown in the next section.

4 Experimental Results

The experiments were done just on two problem instances, $P = 5, N = 8$ and $P = 6, N = 9$. These are difficult enough to be able to make the quality of the different algorithms apart, but also simple enough to perform the experiments in an amount of time that is affordable. The experiments were performed on a published set of 5000 instances and playing a single game for each instance. Experiments for BS and Evo++ were done on different computers. Evo++ used a single parametrization, shown in table 1 except for population and consistent set size. Population and consistent set size are found by testing different values and using rules of thumb (consistent set size should be approximately 1/10 of population size). However, the results in this case are not guaranteed to be the best; we stopped when we found the best average number of moves, but from that point population and consistent set size (and thus number of evaluations) could be improved.

Parameters for the BS approach are shown in Table 2. These parameters are the same for the Search and Decision EAs in the BS algorithm. The specific values of these parameters were found through systematic experimentation.

Table 3. Comparison among approaches in this paper: BS, Evo and Berghman et al. [17]

(a) Mean number of guesses with standard deviation; the quantities in parentheses indicate population and consistent set size (in the case of Evo++). The horizontal line indicates significant differences using Wilcoxon paired test. There is also significant difference for $P = 6, N = 9$ between the two versions of Evo++, but not in the other case.

	$P = 5, N = 8$		$P = 6, N = 9$	
Berghman et al.				6.475
Evo++ (EN)	(1000,200)	5.555 ± 0.011	(2000,200)	6.373 ± 0.011
BS		5.518 ± 0.009		6.382 ± 0.011
Evo++ (MP)	(1000,80)	5.602 ± 0.012	(2000,200)	6.436 ± 0.012

(b) Mean number of evaluations with standard deviation; differences are always significant

	$P = 5, N = 8$	$P = 6, N = 9$
Evo++ (EN)	26098 ± 144	67521 ± 384
BS	40619 ± 115	45478 ± 168
Evo++ (MP)	20120 ± 114	68860 ± 406

Two different measures of quality were used: the average number of moves and the number of evaluations. The first is a measure of the quality of the algorithm playing the game, and the second of algorithm performance; being EAs search algorithms, the smaller percentage of the search space that is explored, the better. Results are shown in Table 3.

The first outstanding result, looking at Table 3(a) is that BS and Evo++ are able to obtain roughly the same results (statistically insignificant difference) using Entropy as score, even if the methods to obtain the consistent set, and even the size of the set, is completely different; at the same time, we can say that it is also proved that EN is significantly better than MP at least for these sizes, even if at smaller sizes there is not a significant difference, at least in exhaustive search studies [8]. We could thus affirm that the main factor in optimizing the number of moves is the scoring method.

The conclusion is not so clear regarding the evaluations, and thus the running time it depends on. These are shown in Table 3(b). Evo++ in all versions beats BS for the smaller size. However, its results are much worse for the bigger size $P = 6, N = 9$. In that sense, and taking into account both results, we could say that depending on the size, BS or Evo++ (EN) are the best methods. Evo++ (MP) is always dominated, in both senses, by the other methods. This supports the conclusion, advanced in the abstract, that Entropy is a better scoring strategy than Most Parts; both were proved better than the rest (at least for small sizes) in [18], which leads us to conclude that, to minimize the number of moves needed to obtain the solution, Entropy should be used to score eligible solutions.

5 Conclusions and Future Work

This paper presents a comparison between a novel and two existing evolutionary approaches for the Mastermind game. The novel approach is a hierarchical algorithm that involves two different evolutionary approaches, one for selecting the best eligible set and another one to select the best playable code. We have detailed this new algorithm and carried out an experimental comparison with two existing evolutionary approaches to draw some interesting conclusions on the performance of the evolutionary techniques for the Mastermind.

It is interesting to note that, independently of the method used, the results obtained are quite similar. Evo++(EN) and BS obtain a number of moves that is quite similar, and better (at least a priori and without complete statistical information) than the result published by Berghman [17], who uses Expected Size. This probably leads to the conclusion that the best strategy for scoring eligible solutions in the game of Mastermind is Entropy, at least if the method for searching them is biased towards those with better score, as evolutionary algorithms are.

An interesting conclusion is also that BS does not increase the number of evaluations so steeply as Evo++ does. While the latter almost triples the number of evaluations (and thus the time needed to examine them), BS does not need so many evaluations. However, it remains to be seen their behavior time-wise, since their mechanisms are completely different.

As future lines of work we will try to examine more closely what are the precise mechanisms that make the algorithms behave differently with respect to the number of evaluations needed to find a playable move. It will be also necessary to measure speeds in similar conditions and, finally, to find out the way they behave when taken to more difficult problems ($P = 7, N = 10$).

Acknowledgement. This work has been partially supported by Spanish Ministry of Science and Innovation, under project numbers ECO2010-22065-C03-02. and TIN2011-28627-C04-02 and P08-TIC-03903 awarded by the Andalusian Regional Government.

References

1. Knuth, E.: The computer as Master Mind. Journal of Recreational Mathematics 9, 1–6 (1977)
2. Irving, W.: Towards an optimum Mastermind strategy. Journal of Recreational Mathematics 11(2), 81–87 (1979)
3. Koyama, K., Lai, T.: An optimal Mastermind strategy. Journal of Recrational Mathematics 25(4), 251–256 (1993)
4. Bestavros, A., Belal, A.: Master Mind: a game of diagnosis strategies. In: Bulletin of the Faculty of Engineering, Alexandria University, Alexandria (1986)
5. Kooi, B.: Yet another mastermind strategy. ICGA Journal 28(1), 13–20 (2005)
6. Chen, S.T., Lin, S.S., Huang, L.T.: A two-phase optimization algorithm for Mastermind. The Computer Journal 50(4), 435–443 (2007)

7. Chen, S.T., Lin, S., Huang, L., Hsu, S.: Strategy optimization for deductive games. European Journal of Operational Research 183, 757–766 (2007)
8. Merelo, J.J., Mora, A.M., Cotta, C., Runarsson, T.P.: An experimental study of exhaustive solutions for the Mastermind puzzle. ARXiV (2012)
9. Shapiro, E.: Playing Mastermind logically. SIGART Bulleting 85, 28–29 (1983)
10. Swaszek, P.: The mastermind novice. Journal of Recreational Mathematics 30, 130–138 (2000)
11. Temporal, A., Kovacs, T.: A heuristic hill climbing algorithm for Mastermind. In: Proc. of the UK Workshop on Computational Intelligence, Bristol, UK, pp. 183–196 (2003)
12. Bernier, J., Herráiz, C., Merelo-Guervós, J.J., Olmeda, S., Prieto, A.: Solving Mastermind using GAs and simulated annealing: a case of dynamic constraint optimization. In: Proc. of the 4th International Conference on Parallel Problem Solving from Nature, London, UK, pp. 554–563 (1996)
13. Bento, L., Pereira, L., Rosa, A.: Mastermind by evolutionary algorithms. In: Proc. of the Sixth Annual Workshop on Selected Areas in Cryptography, Kingston, Ontario, Canada, pp. 307–311 (1999)
14. Kalister, T., Camens, D.: Solving Mastermind using Genetic Algorithms. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1590–1591. Springer, Heidelberg (2003)
15. Merelo-Guervós, J.J., Castillo, P., Rivas, V.: Finding a needle in a haystack using hints and evolutionary computation: the case of evolutionary MasterMind. Applied Soft Computing 6(2), 170–179 (2006)
16. Some A. Uthor, A fine paper (2012)
17. Bergman, L., Goossens, D., Leus, R.: Efficient solutions for Mastermind using genetic algorithms. Computers & Operations Research 36(6), 1880–1885 (2009)
18. Runarsson, T.P., Merelo-Guervos, J.J.: Adapting heuristic Mastermind strategies to evolutionary algorithms. In: Proc. of the International Workshop on Nature Inspired Cooperative Strategies for Optimization, Granada, Spain (2010)

A Genetic Algorithm for Color Image Segmentation

Alessia Amelio and Clara Pizzuti

Institute for High Performance Computing and Networking,
National Research Council of Italy, CNR-ICAR,
Via P. Bucci 41C, 87036 Rende (CS), Italy
{amelio,pizzuti}@icar.cnr.it

Abstract. A genetic algorithm for color image segmentation is proposed. The method represents an image as a weighted undirected graph, where nodes correspond to pixels, and edges connect similar pixels. Similarity between two pixels is computed by taking into account not only brightness, but also color and texture content. Experiments on images from the Berkeley Image Segmentation Dataset show that the method is able to partition natural and human scenes in a number of regions consistent with human visual perception. A quantitative evaluation of the method compared with other approaches shows that the genetic algorithm can be very competitive in partitioning color images.

1 Introduction

Image segmentation is an important problem in pattern recognition that aims at dividing an image into a number of regions having high homogeneity inside the same region, while adjacent regions are significantly dissimilar with respect to some adopted homogeneity measure. Many approaches have been proposed to segment monochrome and color images [7, 25, 27]. However, as observed by Cheng et al. [7], color image segmentation techniques are considered more appealing since they can provide more information than grey level images, and human eye is able to better detect objects when color is present. Most of these proposals, as outlined in [7], extend gray level image segmentation methods, such as histogram thresholding, boundary detection, region based approaches, with color representations. Furthermore, it has been recognized that no general algorithm exists for all monochrome and color images. Thus, techniques specialized for particular application domains have been presented [3–6, 12–14].

Among the several techniques proposed for image segmentation, methods representing an image as a graph [9, 20, 23, 24, 26], in which nodes correspond to pixels, and an edge between two pixels exists if they are similar, on the base of a suitably defined similarity criterion, revealed competitive both in terms of efficiency and segmentation quality [9]. In particular, Shi and Malik [20] introduced the concept of *normalized cut* that allows the partitioning of a graph in groups of nodes such that the homogeneity inside each region is maximized, while minimizing the dissimilarity between regions. More recently, Maji et al. [16] proposed the *biased normalized cut*, a modification of the normalized cut to incorporate priors which can be used for constrained color-texture based image segmentation.

In this paper a genetic algorithm for color image segmentation that adopts the representation of an image as a graph is proposed. The algorithm, named *C-GeNCut* (*Color*

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 314–323, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Genetic Normalized Cut), extends the approach proposed in [1] to take into account color, brightness, and texture, by computing the affinity between two pixels with the *Intervening Contour* cue [8, 10, 11, 15], that uses the *multispectral Pb detector* as defined by Arbelaez et al. in [2]. Experiments on ten color images from the Berkeley Image Segmentation Dataset (BSDS300) [17] show that the method is able to partition natural and human scenes in meaningful objects. A quantitative evaluation based on the well known concept of *Probabilistic Rand Index*, defined by Unnikrishnan et al. [21, 22], shows that the inclusion of color and texture improves the segmentation accuracy with respect to the algorithm of Amelio and Pizzuti [1], that considers only the gray-level information, as well as with the segmentations obtained by the algorithm of Maji et al. [16] for color images.

The paper is organized as follows. In the next section the problem of image segmentation is defined, together with its formalization as a graph partitioning problem, and a description of the homogeneity measure adopted. Section 3 describes fitness function, the genetic representation, and operators employed. Section 4 describes the evaluation measure used. Section 5 presents the experimental results. Finally, section 6 summarizes the approach.

2 Graph-Based Segmentation

An image R can be represented as a weighted undirected graph $G = (V, E, w)$, where V is the set of the nodes, E is the set of edges in the graph, and $w : E \rightarrow \mathcal{R}$ is a function that assigns a value to graph edges. Each node corresponds to a pixel in the image, and an edge (i, j) connects two pixels i and j, provided that these two pixels satisfy some property suitably defined that takes into account both pixel characteristics and spatial distance. The weight $w(i, j)$ associated with a graph edge (i, j) represents the likelihood that pixels i and j belong to the same image region and provides a similarity value between i and j. The higher the value of $w(i, j)$, the more likely the two pixels are members of the same region. Let W be the adjacency weight matrix of the graph G. Thus W_{ij} contains the weight $w(i, j)$ if the nodes i and j are connected, zero otherwise. Depending on the method adopted to compute the weights, any two pixels may or may not be connected.

2.1 Affinity Computation

In order to compute the weights, differently from [1], in this approach we employed the *Intervening Contour* cue [8, 10, 11, 15] that uses the multispectral *Pb* detector as defined in [2]. In this framework, given a generic pixel, the value of the multiscale *Pb* detector at that pixel is considered. If the maximum value along a straight line connecting the two pixels i and j in the image plan is large, then a deep change and, consequently, an intervening contour is present, indicating that the two pixels don't belong to the same segment. Hence, the weight $w(i, j)$ between these pixels will be low. On the other hand, if the value of the multiscale *Pb* detector is sufficiently weak, this usually happens in a region that is flat based on brightness, color and texture, the affinity between the two

pixels will be very high. More formally, the weight $w(i,j)$ between the pixels i and j is computed as:

$$w(i,j) = \begin{cases} e^{-max_{p \in line(i,j)}\{mPb(p)\}/\rho} & \text{if } ||X(i) - X(j)||_2 < r, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

where *line(i,j)* is a straight line between i and j, *X(i)* is the spatial location of the pixel i, *r* is a distance threshold and ρ is a constant.

Multiscale *Pb* detector is based on the *Pb* detector function $Pb(x, y, \theta)$. Given an image pixel at position (x, y), it represents the posterior probability of a boundary with orientation θ at that pixel. This measure is obtained by evaluating the difference in local image brightness, color and texture channels.

Specifically, input image is transformed into four distinct channels. The first three channels are those of the CIE Lab colorspace: brightness, color a and color b. Color a represents the position of the color between red/magenta and green, while color b indicates the position of the color between yellow and blue. The last channel is related to the image texture content and it assigns to each pixel a texton id. Associations between pixels and texton ids come from another previous filtering stage. In particular, the input image is converted to grayscale and processed by a set of 17 Gaussian derivative and center-surround filters. Consequently, each pixel is represented by a 17-dimensional vector of responses, composed of one value from each filter. After that, these vectors are clustered by using $k - Means$: the cluster centers identify the image textons and each pixel is associated with the id in $[1, k]$ of the closest cluster center. Experiments provided 32 as a sufficient value for k. Finally, the texton channel is built, where each pixel of the original image is substituted by its corresponding texton id.

For each image channel, an oriented gradient signal $G(x, y, \theta)$ is computed at position (x, y), by placing a circular disc centered at location (x, y) and splitting it into two half-discs g and h by a diameter at angle θ. For each half-disc, an histogram of the intensity values of the pixels covered by it, is built. The gradient magnitude G at location (x, y) is defined by the χ^2 distance between the two half-disc histograms g and h.

$$\chi^2(g,h) = \frac{1}{2} \sum_i \frac{(g(i) - h(i))^2}{g(i) + h(i)}$$

Furthermore, gradients at three scales $[\sigma/2, \sigma, 2\sigma]$ are considered for each channel, in order to detect fine and coarse image features.

The *Pb* detector processes the channels separately and then combines the oriented gradient signals obtained from the different channels at multiple scales into a single multiscale oriented signal:

$$mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta)$$

where s represents the scales index, i the feature channel index (brightness, color a, color b and texture) and $G_{i,\sigma(i,s)}(x, y, \theta)$ the oriented gradient signal at (x, y) in channel i where the radius of the disc is $\sigma(i, s)$ and the angle is θ. The parameters $\alpha_{i,s}$ weight the contribution of each gradient signal. The angle θ defining the orientation, takes eight

different values in the interval $[0, \pi)$. The final value of the multispectral *Pb* detector is the maximum response over the eight orientations:

$$mPb(x, y) = max_\theta\{mPb(x, y, \theta)\}$$

3 Algorithm

In this section we briefly describe the genetic operators and fitness function proposed in [1] and adopted also for *C-GeNCut*. The representation of individuals is based on the locus-based adjacency representation proposed in [18]. In this graph-based representation an individual of the population consists of N genes $g_1, \ldots, g_N$ and each gene can assume allele values j in the range $\{1, \ldots, N\}$. Genes and alleles represent nodes of the graph $G = (V, E, w)$ modelling an image, and a value j assigned to the ith gene is interpreted as a link between the pixels i and j. The initialization process assigns to each node i one of its neighbors j, and the kind of crossover operator adopted is uniform crossover. The mutation operator randomly assigns to each node i one of its neighbors. For both initialization and mutation, an important aspect to consider is the determination of the neighbors of each node. The concept of neighbors of a node introduced in [1] takes into account not only the spatial closeness but also the pixel affinity. More in details, given a generic node i in the graph, let $w^h_{max} = \{w^1, \ldots, w^h \mid w^1 \geq, \ldots, \geq w^h\}$ be the first h highest weights of row i in the weight adjacency matrix W.

The h nearest neighbors of i, denoted as nn^h_i, are then defined as $nn^h_i = \{j \mid w(i, j) \in w^h_{max}\}$. nn^h_i is thus the set of those pixels that are no more than r pixels apart from i, and that have maximum similarity with i. It is worth to note that, even if h is fixed to 1, the number of nearest neighbors of i could be sufficiently large if many of its spatial neighbors have the same maximum weight w^h_{max}. This definition of nearest neighbors guaranties to choose the most similar neighbors during the initialization process, and to bias the effects of the mutation operator towards the most similar neighbors, thus it contributes to improve the results of the method.

The fitness function is an extension of the concept of normalized cut of Shi and Malik [20]. Let $G = (V, E, w)$ be the graph representing an image, W its adjacency matrix, and $P = \{R_1, \ldots, R_k\}$ a partition of G in k clusters.

For a generic cluster $R \in P$, let

$$c_r = \textstyle\sum_{i \in R, j \notin R} W_{ij} \qquad m_r = \textstyle\sum_{i \in R, j \in R} W_{ij} \qquad m = \textstyle\sum_{i \in V, j \in V} W_{ij}$$

be respectively the sum of weights of edges on the boundary of R, the sum of weights of edges inside R, and the total graph weight sum. The *weighted normalized cut WNCut* measures for each cluster in P the fraction of total edge weight connections to all the nodes in the graph

$$WNCut = \sum_{r=1}^{k} \frac{c_r}{m_r + c_r} + \frac{c_r}{(m - m_r) + c_r}$$

Because of the affinity measure w defined in the previous section, more uniform regions can be obtained with low cut values between the subgraphs representing the regions and the rest of the graph. This implies that low values of $WNcut$ are preferred.

4 Evaluation Measure: Probabilistic Rand Index

In the Berkeley dataset, for each image, multiple human-traced segmentations for color images are available. All the segmentations are considered equally reliable. As observed in [22], when multiple ground-truth segmentations are available for the same image, the comparison should be made against all the manually obtained segmentations. To this end, Unnikrishnan et al. [21, 22] introduced the *Probabilistic Rand Index* as an extension of the concept of *Rand Index* [19], employed to assess clustering methods. Given a set $\{S_1, \ldots, S_T\}$ of ground-truth segmentations of an image I consisting of n pixels, and a test segmentation S, the *Probabilistic Rand Index* is defined as :

$$PRI(S, \{S_1, \ldots, S_T\} = \frac{1}{H} \sum_{i<j} [c_{ij} p_{ij} + (1 - c_{ij})(1 - p_{ij})]$$

where c_{ij} denotes the event that pixels i and j have the same label, p_{ij} is its probability, and $H = n*(n-1)/2$ is the total number of pixel pairs. The PRI value varies between 0 and 1. When its value is 0 it means that S and $\{S_1, \ldots, S_T\}$ are completely dissimilar.

5 Experimental Results

In this section we present the results of *C-GeNCut* on ten images from the Berkeley Image Segmentation Dataset (BSDS300) [17] and compare the performances of our algorithm in partitioning natural and human scenes in meaningful objects with the segmentations obtained by the algorithm of Maji et al. [16] (*Biased NCut*) for color images, in the following referred as *C-NCut*, and by the algorithm of Amelio and Pizzuti [1] (*GeNCut*), that takes into account only grayscale information, on the same images.

The version of the *Biased NCut* software is written in MATLAB and it is available at http://ttic.uchicago.edu/ smaji/projects/biasedNcuts/. However we eliminated the interactive mode from the available algorithm specifically for performing comparisons with our technique.

The *C-GeNCut* algorithm has been written in MATLAB 7.14 R2012a, using the Genetic Algorithms and Direct Search Toolbox 2. In order to set parameter values, a trial and error procedure has been employed and then the parameter values giving good results for the benchmark images have been selected. Thus we set crossover rate to 0.9, mutation rate to 0.2, elite reproduction 10% of the population size, roulette selection function. The population size was 100, the number of generations 60. The value h of nearest neighbors to consider has been fixed to either 1 or 2. As already pointed out, this does not mean that the number of neighbors is 1 or 2, but that the first (and second) most similar neighbors are taken into account for the initialization and mutation operators. The fitness function, however, is computed on the overall weight matrix. For all the data sets, the statistical significance of the results produced by *C-GeNCut* has been checked by performing a t-test at the 5% significance level. The p-values returned are very small, thus the significance level is very high since the probability that a segmentation computed by *C-GeNCut* could be obtained by chance is very low.

The weight matrix of each image is computed in the same way for both *C-NCut* and *C-GeNCut* methods, and, as already described in section 2, it is based on the Intervening

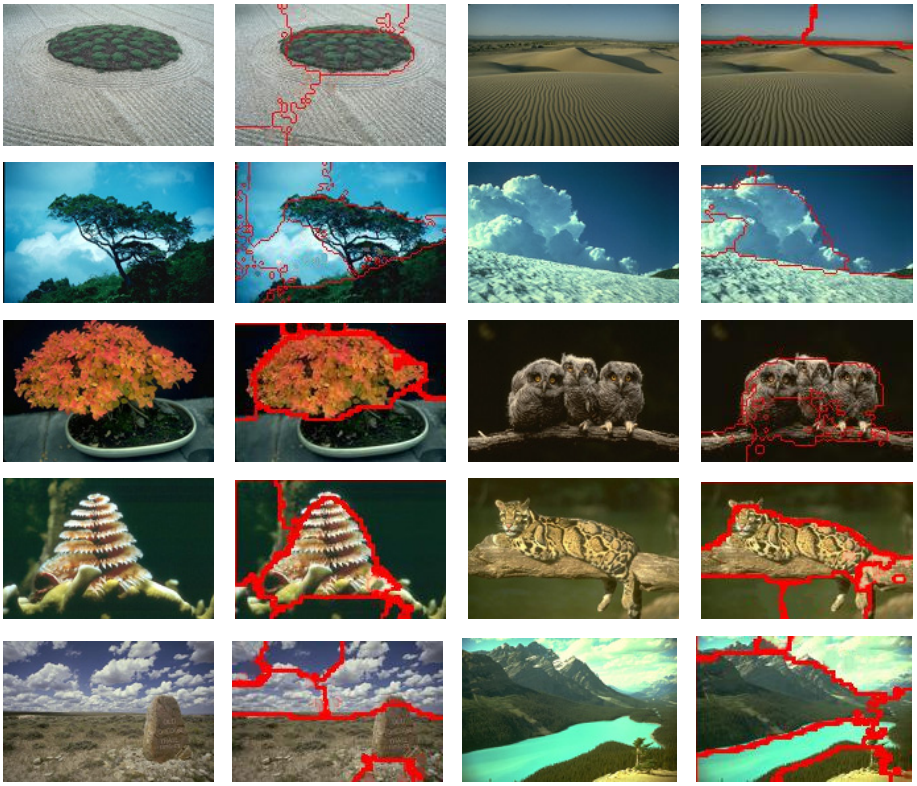


Fig. 1. Segmentation of *C-GeNCut* on ten images of the Berkeley Image Segmentation dataset (BSDS300). For each image, the original version together with the segmentation result of *C-GeNCut* are presented.

Contour framework that uses the multiscale *Pb* detector by fixing $r = 5$ and $\rho = 0.1$. About the *Pb* detector, the parameter σ, which defines the scales, is set to 5 pixels for the brightness channel, while for color and texture channels σ is set to 10 pixels. The parameters $\alpha_{i,s}$ are fixed to $0.01, 0.01, 0.02, 0.02, 0.02, 0.03, 0.02, 0.02, 0.02, 0.01, 0.01$ and 0.01. The weight matrix of each image is computed for *GeNCut* as in [1], by fixing $r = 10$, number of scales 3, number of orientations 4 and $\sigma = 0.1$. The value h of nearest neighbors to consider has been fixed to either 1 or 2.

In order to compare *C-GeNCut* and *C-NCut*, given an image I, we executed *C-NCut* as many times as the different number of segmentations available for I, by giving as input the distinct values k of ground-truth segments corresponding to the color human-segmentations. This implies that *C-NCut* has been executed for the best input parameter value k. For each image and for each distinct number of segments from the ground-truth segmentations, *C-NCut* has been run 10 times. The average values of the Probabilistic Rand Index (PRI) [21] have been computed, together with the standard deviation, for the partitioning found by *C-NCut* and *GeNCut*, and compared with that obtained

Table 1. Probabilistic Rand Index evaluated for *C-NCut*, *C-GeNCut* and *GeNCut* on a subset of the Berkeley Image Segmentation Dataset (BSDS300). PRI represents the Probabilistic Rand Index, nc the number of segments.

	GeNCut		***C-GeNCut***		***C-NCut***	
	nc	PRI	nc	PRI	nc	PRI
I_1	11	0.6443 (0.0637)	5	0.7526 (0.0263)	5	0.7299 (0.0003)
					24	**0.7858** (0.0008)
					4	0.7333 (0.0001)
					23	**0.7862** (0.0003)
					41	**0.7858** (0.0002)
					17	**0.7856** (0.0006)
I_2	8	0.7035 (0.0081)	3	**0.7613** (0.0063)	5	0.6728 (0)
					3	0.7068 (0.0001)
					8	0.6794 (0.0078)
					7	0.6799 (0.0001)
					9	0.6774 (0.0001)
I_3	13	0.7041 (0.0183)	11	**0.7052** (0.0183)	11	0.6545 (0.0001)
					16	0.6446 (0.0001)
					30	0.6202 (0.0007)
					10	0.6651 (0.0017)
					26	0.6235 (0.0009)
I_4	16	0.7889 (0.0368)	5	**0.8339** (0.0213)	7	0.7277 (0.0003)
					3	0.8200 (0.0001)
					5	0.8260 (0.0001)
					8	0.7082 (0.0144)
					6	0.8047 (0.0001)
I_5	13	0.8088 (0.0198)	7	0.8235 (0.0065)	7	0.8168 (0.0001)
					6	**0.8338** (0.0001)
					10	0.7977 (0.0001)
					8	0.8191 (0.0060)
					15	0.7861 (0.0001)
I_6	4	0.8036 (0.0186)	6	**0.8288** (0.0379)	13	0.7565 (0.0001)
					6	0.7455 (0.0003)
					19	0.7405 (0.0001)
					23	0.7405 (0.0013)
I_7	9	0.7308 (0.0118)	6	**0.7820** (0.0101)	6	0.7512 (0.0018)
					4	0.7345 (0.0001)
					10	0.7101 (0.0062)
					2	0.5763 (0)
I_8	10	0.8215 (0.0002)	5	**0.8361** (0.0163)	5	0.7827 (0.0001)
					8	0.8217 (0.0001)
					7	0.8122 (0.0001)
					10	0.8109 (0.0012)
I_9	18	0.7425 (0.0059)	6	**0.7653** (0.0076)	6	0.7399 (0.0001)
					5	0.7352 (0.0001)
					3	0.7114 (0.0001)
					9	0.7212 (0.0001)
					8	0.7041 (0.0001)
					28	0.6825 (0.0009)
I_{10}	8	0.7797 (0.0375)	8	0.8361 (0.0075)	39	0.8339 (0.0005)
					10	**0.8557** (0.0001)
					8	**0.8446** (0.0001)
					15	**0.8471** (0.0001)
					26	**0.8405** (0.0001)
					24	**0.8447** (0.0001)

by *C-GeNCut* on the same images. Since *C-GeNCut* and *GeNCut* generate a single segmentation, the average value of PRI is the same for each of the values of ground-truth segments of the image under consideration.

Table 1 reports the PRI for *C-GeNCut*, *C-NCut* and *GeNCut*. In particular, for each image I we computed the PRI value of the segmentation returned by *C-GeNCut* and by *GeNCut*, considered as test segmentation, against the set of ground-truth segmentations associated with I in the Berkeley dataset. As regards *C-NCut*, the PRI values have been computed by considering as test segmentation that obtained by *C-NCut* for each of the executions performed, i.e. one for the input parameter k fixed to the number of segments obtained by *C-GeNCut*, and one for every distinct value of ground-truth segments available for the image under consideration.

For example, if we consider image I_1, for which five human segmentations are available, *C-GeNCut* found a segmentation of 5 segments with PRI value equal to 0.7526, while *GeNCut* obtained 11 regions and PRI value 0.6443. The PRI values for *C-NCut* are 0.7299, 0.7858, 0.7333, 0.7862, 0.7858, 0.7856 when as test segmentations are used those obtained for input parameter k equal to 5, which is the number of segments returned by *C-GeNCut*, and 24, 4, 23, 41, and 17, respectively, that correspond to the distinct number of segments from the ground-truth segmentations. Note that, for images I_2, I_4, I_6, I_7, and I_{10} *C-GeNCut* found a number of segments equal to one of the ground-truth segmentations.

The table points out that the PRI value of *C-GeNCut* is always higher than the corresponding PRI value of *GeNCut*. Furthermore, the PRI value of *C-GeNCut* is the highest for seven out of the ten images, i.e. for I_2, I_3, I_4, I_6, I_7,I_8, and I_9, also with respect to *C-NCut*. As regards the other three images, *C-NCut* overcomes *C-GeNCut* on I_1 for 4 out of 6 segmentation values, on I_5 for 1 out of 5 segmentation values, and on I_{10} for 5 out of 6 segmentation values. *C-GeNCut* thus improves the results of the genetic approach when color and texture information are included, and it is competitive with respect to *C-NCut*, that is specialized for color images.

Finally, in Figure 1, for each of the ten images, we present the segmentation outputs of *C-GeNCut* by depicting the contours of the regions on the original image. The visual perception of the segmentation results is quite positive: the main objects of a scene are identified and the most meaningful features extracted from the images by the segmentation process.

6 Conclusions

The paper presented a graph-based approach to image segmentation that employs genetic algorithms. The method extends the method proposed in [1], by considering not only brightness but also color and texture for image segmentation. The method revealed particularly apt to deal with color-texture images modeled as graphs. In fact, as experimental results showed, the genetic approach is able to segment color-texture images in a number of regions that well adhere to the human visual perception, and it is competitive with state-of-the-art methods for color image segmentation.

Acknowledgements. This work has been partially supported by the project *MERIT : MEdical Research in Italy*, funded by MIUR.

References

1. Amelio, A., Pizzuti, C.: An Evolutionary and Graph-Based Method for Image Segmentation. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 143–152. Springer, Heidelberg (2012)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Transactions on Pattern Anal. Mach. Intell. 33(5), 898–916 (2011)
3. Ballerini, L., Bocchi, L., Johansson, C.B.: Image Segmentation by a Genetic Fuzzy c-Means Algorithm Using Color and Spatial Information. 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. 260–269. Springer, Heidelberg (2004)
4. Bevilacqua, V., Mastronardi, G., Piazzolla, A.: An Evolutionary Method for Model-Based Automatic Segmentation of Lower Abdomen CT Images for Radiotherapy Planning. 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, Part I. LNCS, vol. 6024, pp. 320–327. Springer, Heidelberg (2010)
5. Chaabane, S.B., Sayadi, M., Fnaiech, F., Brassart, E.: Dempster-shafer evidence theory for image segmentation: Application in cells images. International Journal of Information and Communication Engineering 5(2), 126–132 (2009)
6. Chen, C.W., Luo, J., Parker, K.J.: Image segmentation via adaptive k-means clustering and knowledge-based morphological operations with biomedical applications. IEEE Transactions on Image Processing 7(12), 1673–1683 (1998)
7. Cheng, H., Jiang, X., Sun, Y., Wang, J.: Color image segmentation: advances and prospects. Pattern Recognition 34, 2259–2281 (2001)
8. Cour, T., Bénézit, F., Shi, J.: Spectral segmentation with multiscale graph decomposition. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)), pp. 1124–1131 (2005)
9. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision 59(2), 167–181 (2004)
10. Fowlkes, C., Malik, J.: How much does globalization help segmentation. Tech. rep. (2004)
11. Fowlkes, C., Martin, D., Malik, J.: Learning Affinity Functions for Image Segmentation: Combining Patch-based and Gradient-based Approaches. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2 (2003)
12. Ghosh, P., Mitchell, M.: Segmentation of medical images using a genetic algorithm. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006, pp. 1171–1178. ACM (2006)
13. Harrabi, R., Braiek, E.B.: Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images. Eurasip Journal of Image and Video Processing 11 (2012)
14. Kim, S.-M., Kim, W.: An Algorithm for Segmenting Gaseous Objects on Images. 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. 322–328. Springer, Heidelberg (2004)
15. Leung, T., Malik, J.: Contour Continuity in Region Based Image Segmentation. In: Burkhardt, H.-J., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1406, pp. 544–559. Springer, Heidelberg (1998)
16. Maji, S., Vishnoi, N.K., Malik, J.: Biased normalized cuts. In: CVPR, pp. 2057–2064. IEEE (2011)

17. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision, vol. 2, pp. 416–423 (2001)
18. Park, Y., Song, M.: A genetic algorithm for clustering problems. In: Proc. of 3rd Annual Conference on Genetic Algorithms, pp. 2–9 (1989)
19. Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66, 846–850 (1971)
20. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
21. Unnikrishnan, R., Hebert, M.: Measures of similarity. In: Seventh IEEE Workshops on Application of Computer Vision, WACV/MOTIONS 2005, vol. 1 (2005)
22. Unnikrishnan, R., Pantofaru, C., Hebert, M.: Towards objective evaluation of image segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(6), 929–944 (2007)
23. Urquhart, R.: Graph theoretical clustering based on limited neighborhood sets. Pattern Recognition 15(3), 173–187 (1982)
24. Wu, Z., Leahy, R.: An optimal graph theoretic approach to data clustering: Theory and applications to image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(11), 1101–1113 (1993)
25. Xu, Y., Olman, V., Uberbacher, E.C.: A segmentation algorithm for noisy images: Design and evaluation. Pattern Recognition Letters 19, 1213–1224 (1998)
26. Zahn, C.T.: Graph theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers 20(1), 68–86 (1971)
27. Zhang, Y.: Evaluation and comparison of different segmentation algorithm. Pattern Recognition Letters 18, 963–974 (1997)

Multiobjective Projection Pursuit for Semisupervised Feature Extraction

Mihaela Elena Breaban

Faculty of Computer Science, Al. I. Cuza University, Iasi, Romania
pmihaela@infoiasi.ro

Abstract. The current paper presents a framework for linear feature extraction applicable in both unsupervised and supervised data analysis, as well as in their hybrid - the semi-supervised scenario. New features are extracted in a filter manner with a multi-modal genetic algorithm that optimizes simultaneously several projection indices. Experimental results show that the new algorithm is able to provide a compact and improved representation of the data set. The use of mixed labeled and unlabeled data under this scenario improves considerably the performance of constrained clustering algorithms such as constrained k-Means.

1 Introduction

Feature extraction plays several key roles in data analysis. Firstly, it aims at providing a reduced representation of the data set in order to visualize data or lower the computational cost for further analysis. Secondly, it extracts explanatory variables that highlight the presence of groups in clustering (also named unsupervised classification) or predict with high accuracy the class labels (classification) or a numerical outcome (regression). Feature extraction can serve more specific goals, as extracting independent signals from data.

The current paper proposes a framework for feature extraction in the context of classification. It covers the supervised case when the data is labeled, the unsupervised case when labels are not provided but groups must be uncovered in data (clustering) and the mixed scenario when only part of the cases are labeled while some are not. The last scenario is known as semisupervised learning/classification and is generally approached from two perspectives: 1) clustering algorithms are guided by a few constraints expressed as pairs of items that must lie in the same cluster and pairs of items that must lie in distinct clusters and 2) classifiers are provided unlabeled data in order to refine their decision boundary.

Our framework draws its roots from classical Projection Pursuit (PP). Such an approach for feature extraction benefits from reduced computational complexity relative to wrapper scenarios. It allows us to perform feature extraction in a filter manner: the evaluation of a new feature is based on a projection index and does not require expensive supervised/unsupervised classification algorithms to assess the quality of the new features as wrapper methods do.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 324–333, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Our framework performs a multi-modal search and simultaneously optimizes two (possible conflicting) objectives, delivering in one run several new features that correspond to an improved multi-dimensional representation of the data set.

The paper is structured as follows. Section 2 describes the traditional framework of projection pursuit and section 3 shortly reviews some PP indices. Section 4 describes the multi-objective framework we propose to extract a new meaningful representation of data. Section 6 concludes the paper with a short discussion and future directions.

2 Projection Pursuit

Projection Pursuit (PP) is essentially an exploratory data analysis technique. It generally aims to deliver low-dimensional linear projections of a data set that reveal interesting properties. Its applicability ranges from simple tasks as data visualization, to assisting unsupervised learning by discovering clusters in data, or supervised learning by deriving linear combinations of attributes that discriminate between given classes and predict a numerical variable.

A k dimensional projection of a data set $X \in R^{n \times d}$ consisting of n items described by d numerical attributes is a linear transformation involving k orthogonal vectors in a d-dimensional space. These vectors form an orthogonal basis $A \in R^{k \times d}$. The projection of X into A is the product $Z = X \cdot A^T$ resulting in a new representation for each of the n data items in a k-dimensional space.

PP originates in the work of Kruskal [4] but the term was introduced later by Friedman and Tukey [6] and generally designates any method that derives linear projections of data that present some meaningful structures.

A PP method mainly consists of two components:

- a projection index to measure the quality of the projection;
- an optimization algorithm to deliver linear combinations that optimize the projection index.

The first component is formulated in accordance with the data analysis task. For simple visualization or dimensionality reduction, random projections can be used or ones that maximize the variance (Principal Component Analysis), the distance from Gaussianity or the Mutual Information (Independent Component Analysis), an index that measures the grouping tendency or indicates the presence of outliers (Friedman's index or Kurtosis). If the main task is data classification, Fisher's criterion can be used, leading to the well-known classification method named Linear Discriminant Analysis.

The second component is closely related to the index under optimization. In isolated cases analytical methods exist (the case of PCA). If the index is differentiable, gradient-based methods are used, otherwise general-applicable probabilistic heuristics can be used.

3 PP Indices for Unsupervised and Supervised Classification

In order to detect clusters in data, an index able to measure the grouping tendency must be used. Several indices exist in literature with this goal. Most of them are formulated for identifying only one-dimensional projections of data; these are faster to compute and easier to optimize. One-dimensional projection indices for clustering are based on computing the entropy or higher-order moments [8].

The current work makes use of one-dimensional projection indices that have low computational complexity.

Kurtosis for a one-dimensional projection y of n data items is defined as the fourth moment around the mean divided by the square of the variance:

$$kurt(z) = \frac{1}{n-1} \sum_{l=1}^{n} \frac{(z^{(l)} - \mu)^4}{\sigma^4} \tag{1}$$

where μ is the mean and σ is the standard deviation of the single-dimensional projection. A value close to 3 indicates a normal distribution. Higher values indicate the presence of extreme deviations while lower values indicate bimodal distributions. We choose to minimize this index mainly because of its reduced computational complexity compared to other proposed indices for cluster detection. When the projection is normalized to mean 0 and variance 1 the index consists in summing up the values raised to the fourth power, divided by the total number of items minus 1.

The *holes* index [2] is maximized for projections that have a hole in the center, corresponding also to bimodal distributions:

$$holes(z) = \frac{1}{n} \sum_{l=1}^{n} e^{(-z^{(l)})^2} \tag{2}$$

There also exist indices that address two-dimensional or higher dimensional projection spaces [5] for clustering. Generally, indices based on the distribution of the distances between data items can be employed for multi-dimensional projections, but these are computationally expensive.

Regarding supervised classification, the most popular PP algorithm is Linear Discriminant Analysis (LDA). Originally introduced by Fisher in 1936, LDA builds a classifier by searching linear projections that best discriminate between given classes. The index widely used in this context is known as Fisher's criterion and, for any two classes C_1 and C_2 it computes the ratio between the inter-class variance and the sum of the intra-class variances:

$$F(z) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{3}$$

Where
$m_i = \frac{1}{|C_i|} \sum_{z^{(l)} \in C_i} z^{(l)}$ is the mean of class C_i,
and
$s_i^2 = \frac{1}{|C_i|} \sum_{z^{(l)} \in C_i} (z^{(l)} - m_i)^2$ is the variance within class C_i.

It can be generalized to multi-class classification with k classes as follows:

$$F(z) = \frac{\sum_{i=1}^{k} |C_i|(m_i - m)^2}{\sum_{i=1}^{k} s_i^2} \tag{4}$$

Where
$m = \frac{1}{|S|} \sum_{z^{(l)} \in S} z^{(l)}$ is the mean of all data items.

4 Multiobjective PP

Projection Pursuit is essentially a multi-modal problem: distinct projections may highlight various patterns in data. As highlighted by several authors [6], projection pursuit analysis should offer several interesting projections, not only the one that corresponds to the global maximum with regard to the projection pursuit index. The most frequent approach reported in literature with this aim is to use local optimization methods like gradient-based (requiring differentiable indices), hill climbing or simulated annealing with several different initializations [6,7,8]. Also structure removal is proposed once interesting projections are found by conducting the search for subsequent orthogonal projections [6] or by transforming the data along the interesting direction into standard normal [5]. Only recently multi-modal search by means of evolutionary algorithms was used to successfully provide several good linear projections [9].

In the current work we design a genetic algorithm that conducts a multi-modal search while optimizing several objectives. Due to the encoding we use it is able at the same time to deal with high dimensional data.

4.1 Solution Encoding

The Genotype: To scale our algorithm to high dimensional data we use an encoding that allows us to incorporate a mechanism of attribute selection while optimizing at the same time the weights of the selected attributes in a linear combination. A candidate solution corresponding to a chromosome in population thus consists of two parts:

- a boolean string $b \in \{0,1\}^d$ of length equal to the number of attributes d; the string is constrained to have at list one bit set;
- a vector $w \in R^d$ in the d-dimensional Euclidean space (a string of length d of real numbers) playing the role of the projection axis, corresponding in fact to numerical weights in the associated linear transformation.

The Phenotype: The projection $z^{(l)} \in R$ of an item $x^{(l)} \in R^d$ in the subspace encoded by a chromosome is computed as follows: $z^{(l)} = \sum_{i=1}^{d} b_i \cdot w_i * x_i^{(l)}$.

Unselected attributes in a linear combination correspond to 0 weight attributes. This simple trick speeds-up the convergence for high-dimensional spaces: irrelevant attributes do not consume resources while their weights would converge to 0.

4.2 Multi-modal Search along Several Objectives

In order to exploit several good attribute subspaces for optimal projections we use the Multi Niche Crowding GA (MNC-GA) [10], an algorithm able to maintain subpopulations within different niches, to maintain diversity throughout the search and to converge to multiple local optima. The multi-modal approach is necessary in order to deliver several good solutions/projections when our framework is used for unsupervised or supervised classification and is further extended to allow multi-objective optimization when several "disagreeing" evaluation functions are used in the semi-supervised scenario of classification.

MNC-GA is a steady state algorithm that implements replacement based on pairwise comparisons. The few parameters involved can be very easily fine-tuned. Furthermore, its authors offer in-depth mathematical results that show the dynamic of the population and offer guidelines about the parameter values to be used in order to achieve the desired niching pressure during a run.

Both the selection and replacement operators implement a crowding mechanism. Mating and replacement within members of the same niche are encouraged while allowing at the same time some competition for the population slots among the niches.

Selection for recombination takes place in two steps: one individual is selected randomly from the population; its mate is the most similar individual from a group of size s which consists of randomly chosen individuals from the population. The two chosen individuals are subject to recombination operators and one offspring is created. The similarity between two individuals is computed based on the boolean part of the chromosome that encodes the monomial subspace; the Hamming distance is used. Recombination between individuals that encode similar subspaces (selected attributes) is necessary in order to conduct a meaningful search in the space of numerical weights.

Regarding **selection for survival**, the individual to be replaced by the offspring is chosen according to a replacement policy called *worst among most similar*: f groups are created by randomly picking g (crowding group size) individuals per group from the population and one individual from each group that is most similar to the offspring is identified. This phase results in f individuals similar to the offspring. One of them must be substituted by the offspring. While all other decisions are taken randomly or are based on measuring similarities, this is the step where decisions involve the fitness of the individuals. Performing multi-criteria optimization, replacement is performed based on the concept of Pareto dominance. The substitution takes place only if there exist a solution among the selected ones that is dominated by the offspring with regard to all

objectives. If several solutions are dominated, the solution to be substituted is the one with the lowest fitness under the supervised criterion (Fisher index).

4.3 Variation Operators

Recombination between two chosen individuals consists of crossover that generates one offspring which is subsequently mutated.

The crossover operator applied to two chromosomes consists in fact of two operations performed independently on the two parts of the chromosomes. Uniform crossover is used on the binary segment encoding the monomial subspace. On the numerical segment, the offspring is obtained as a convex combination of the two numerical parent strings.

Mutation is also applied in two distinct phases. Each gene in the binary segment is flipped with a given probability which we call binary mutation rate. To each weight in the numerical segment corresponding to a selected monomial a random value in the interval (-0.25, 0.25) is added with a probability called weights' mutation rate. The binary mutation rate is lower than the weights' mutation rate in order to encourage better exploitation of a given subspace for optimal projection axes.

4.4 Evaluation

Before evaluation, the weight vector in the selected subspace of the offspring is normalized to unit length. The evaluation consists in computing the projection of all data items on the axis given by the weight segment of the chromosome in the encoded monomial subspace, followed by the computation of the projection indices employed as objectives.

Usually, projection pursuit is preceded by a linear transformation on the data called sphering that guarantees that every linear projection is distributed with mean 0 and standard deviation 1, eliminating the need for further normalization. The data sphering achieves invariance of the projection indices. In our experiments the data sets are not sphered but we achieve this invariance by normalizing each projection prior to computing the projection indices.

In our experiments we perform two-criterion optimization with one unsupervised index in order to highlight groups in data and one supervised index in order to extract features that maximize the separation between known classes of data items.

4.5 The Solutions

The algorithm returns the set of non-dominated solutions at the end of the run. These constitute projection axis in a d-dimensional space that are used to deliver a new representation of the data. They are not orthogonal vectors but an orthogonal basis can be constructed based on simple Gram-Schmidt transforms. By projecting the data onto the new axes new features are extracted that can be used further in clustering or classification or in semi-supervised analysis.

Our framework is generally applicable. When no labels are provided, the algorithm optimizes the unsupervised criterion and delivers projection axes that reveal groups in data. With respect to the concept of Pareto-dominance, only one solution is returned. However, because we employ a multi-modal algorithm the entire population of candidate solutions in the last iteration of the algorithm can be processed to extract several local optima with regard to the projection index.

For the case when all data labels are available, the unsupervised objective delivers new projection axes that do not alter the axes maximizing the supervised criterion. If considered, they can increase the generalization ability of the classifier.

5 Experiments

Our framework for feature extraction is validated on real data sets from UCI Repository [1]. To compare our results with previous work in literature on feature selection, some data sets are modified as in [1,3] in order to contain equal-sized classes.

As projection indexes we use Kurtosis and Fisher's index. The set of non-dominated solutions returned at the end of the run are used to compute new features as linear combinations of the original ones. The new features are then used in clustering performed with the traditional k-Means algorithm. As measures of performance, the Adjusted Rand Index (ARI) and the error rate (1-accuracy) are computed.

All experiments were carried out with the following parameter settings: $s = 0.15 \cdot popsize$, $g = 0.10 \cdot popsize$, $f = 0.15 \cdot popsize$. The mutation operator is applied at different rates on the two segments of a chromosome: approximately one mutation during 10 iterations is applied on the binary segment while 1 mutation per iteration is applied on the numerical segment; using a steady-state scheme, only one offspring is generated and evaluated at each iteration. The population consists of 100 individuals, randomly initialized: on the binary segment 50% of the attributes are selected while on the numerical segment the values are generated in the interval [-1,1]. The results are reported after 20000 fitness evaluations.

Table 1 presents the results obtained with standard k-Means on the original data set and constrained k-Means when 5% of the data set is labeled. The average and standard deviations over 20 runs are reported. A small improvement is obtained with regard to the quality of the partition when labels are provided.

Table 2 presents the results obtained with standard k-Means when a new representation of the data set is obtained with our multi-objective projection pursuit algorithm. When no labels are provided (column *k-Means unsupervised*) only one projection (extracted feature) is used - the best projection optimizing the Kurtosis criterion. A significant improvement is observed over the results in Table 1 showing that unsupervised feature extraction by PP is able to eliminate

[1] `http://archive.ics.uci.edu/ml/`

noise and provide an improved representation of the data set. When labels are provided only two projections are used in k-Means - the best with regard to Kurtosis and the best with regard to Fisher's criterion. The table also shows the average size of the Pareto front extracted at the end of the run. The accuracy of classification is increased in all test cases. All the results presented in Table 2 are also averaged over 20 runs.

The results indicate that feature extraction as a pre-processing step prior to (semi)supervised clustering improves considerably the accuracy of classification. Constrained k-Means in the original feature space is significantly outperformed by k-Means in the optimized feature space.

Figure 1 presents the two-dimensional representation of the data sets obtained under the unsupervised criterion when no data labels are provided. On the final population of the multi-modal GA k-Means was applied to extract 3 clusters. The best individual in each cluster was kept and then the Gram-Schmidt transform was applied to derive a 3-dimensional basis. The data was then projected onto this basis and only the first two projections are illustrated. The figure demonstrates the ability of the algorithm that performs a multi-modal search to extract good multi-dimensional projections in one run.

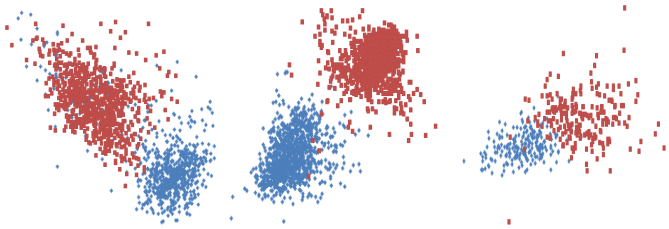
Figure 2 presents the data sets under the new representations corresponding to the two projections extracted when labels are provided. The colors indicate the real labels. Generally, the two projections corresponding to the unsupervised PP index (on the horizontal axis) and respectively the supervised PP index (on the vertical axis) are highly correlated. This observation justifies the use of unlabeled

Table 1. The average ARI and the average error rate for 20 runs for k-Means and constrained k-Means algorithms applied on the original data set

Problem	#items	#features (m)	# classes (k)	k-Means unsupervised ARI	k-Means unsupervised error	k-Means constrained (5%) ARI	k-Means constrained (5%) error
LetterAB	1555	16	2	0.71 ± 0	10.16±5.60	0.73 ±0	7.20 ± 0
satImage	2110	36	2	0.47 ± 0.00	15.69±0.01	0.48 ± 0.003	15.21 ±0.11
WDBC	424	31	2	0.34 ± 0	20.75±0.00	0.37 ±0.005	19.20 ± 0.23

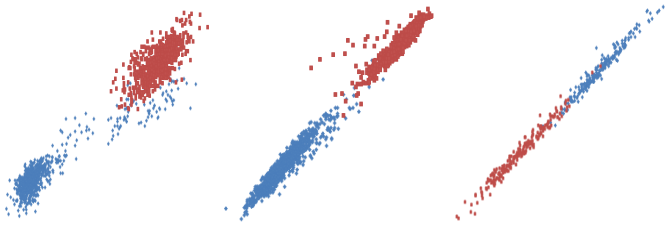
Table 2. The average ARI and the average error rate for 20 runs for k-Means on the new representation of the data set derived with our algorithm. The average number of projection axes (extracted features) in the Pareto front is also reported when labels are provided.

Problem	k-Means unsupervised ARI	k-Means unsupervised error	k-Means 5% #extracted features	k-Means 5% ARI	k-Means 5% error
LetterAB	0.78 ± 0.008	5.77 ± 0.24	37 ± 8	0.79 ± 0.01	5.29 ± 0.39
satImage	0.96 ± 0.01	0.78 ± 0.26	12 ± 4	0.97 ± 0.03	0.70 ± 0.09
WDBC	0.72 ± 0.04	7.39 ± 1.35	24 ± 10	0.78 ± 0.05	5.63 ± 1.57



(a) (b) (c)

Fig. 1. A two dimensional representation of the data sets obtained with our algorithm with the unsupervised PP index and Gram-Schmidt transform: a)Letter AB, b) SAT Image, c)WDBC



(a) (b) (c)

Fig. 2. A two dimensional representation of the data sets obtained with our algorithm with the unsupervised and the supervised PP indices when 5% of the data items are labeled. The best projection axis under the unsupervised criterion(on horizontal) and the best under the supervised criterion (on the vertical) are used. a)Letter AB, b) SAT Image, c)WDBC

data to assist supervised classification algorithms. As expected from the nature of the problem (supervised vs unsupervised classification) and shown by the good correlation between the obtained projections and also by the reduced size of the solutions approximating the Pareto front, the two objectives are not entirely conflicting; however, they disagree on the expected partition.

6 Conclusions

This paper proposes a multi-modal genetic algorithm to extract linear features from data by optimizing several projection indices. The use of an unsupervised index along with a supervised one makes the algorithm general applicable. The current experiments prove the feasibility of the approach on data sets consisting of two classes: the subsequent clustering algorithm achieves better accuracy in the extracted feature space compared to the original feature space. Because kurtosis favors bimodal distributions, future work will extend the framework

with projection indices able to detect an arbitrary number of clusters in data. Also, a new encoding scheme along with multi-dimensional projection indices will be introduced in order to search for multi-dimensional orthogonal projections; these are necessary because the Gram-Schmidt transformation may alter the quality of the projections with regard to the projection indices.

Acknowledgements. This work was supported by POSDRU/89/1.5/S/63663 CommScie grant.

References

1. Breaban, M.E.: Optimized Ensembles for Clustering Noisy Data. In: Blum, C., Battiti, R. (eds.) LION 4. LNCS, vol. 6073, pp. 220–223. Springer, Heidelberg (2010)
2. Cook, D., Buja, A., Cabrera, J.: Projection pursuit indexes based on orthonormal function expansions. Journal of Computational and Graphical Statistics 2(3), 225–250 (1993)
3. Domeniconi, C., Al-Razgan, M.: Weighted cluster ensembles: Methods and analysis. CM Transactions on Knowledge Discovery from Data 2(4) (2009)
4. Friedman, J.H.: Toward a practical method to help uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new 'index of condensation. Statistical Computation 82(397) (1969), Milton, R.C., Nelder, J.A.(eds.)
5. Friedman, J.H.: Exploratory projection pursuit. Journal of the American Statistical Association 82(397), 249–266 (1987)
6. Friedman, J.H., Tukey, J.W.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput. 23(9), 881–890 (1974)
7. Graham, R., Knuth, D., Patashnik, O.D., Cook, Swayane, D.F.: Springer, New York (2007)
8. Jones, M.C., Sibson, R.: What is projection pursuit (with discussion). Journal of the Royal Statistical Society 150, 1–37 (1987)
9. Marie-Sainte, S.L., Berro, A., Ruiz-Gazen, A.: An Efficient Optimization Method for Revealing Local Optima of Projection Pursuit Indices. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 60–71. Springer, Heidelberg (2010)
10. Vemuri, V., Cedeño, W.: Multi-niche crowding for multimodal search. Practical Handbook of Genetic Algorithms: New Frontiers (1995)

Land Cover/Land Use Multiclass Classification Using GP with Geometric Semantic Operators

Mauro Castelli[1,2], Sara Silva[1,3], Leonardo Vanneschi[2,1], Ana Cabral[4], Maria J. Vasconcelos[4], Luís Catarino[4,5], and João M.B. Carreiras[4]

[1] INESC-ID, IST, Universidade Técnica de Lisboa, 1000-029 Lisboa, Portugal
[2] ISEGI, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal
[3] CISUC, Universidade de Coimbra, 3030-290 Coimbra, Portugal
[4] Instituto de Investigação Científica Tropical, 1300-344 Lisboa, Portugal
[5] CIBIO, Universidade do Porto, 4485-661 Vairão, Portugal
sara@kdbio.inesc-id.pt

Abstract. Multiclass classification is a common requirement of many land cover/land use applications, one of the pillars of land science studies. Even though genetic programming has been applied with success to a large number of applications, it is not particularly suited for multiclass classification, thus limiting its use on such studies. In this paper we take a step forward towards filling this gap, investigating the performance of recently defined geometric semantic operators on two land cover/land use multiclass classification problems and also on a benchmark problem. Our results clearly indicate that genetic programming using the new geometric semantic operators outperforms standard genetic programming for all the studied problems, both on training and test data.

1 Introduction

A new integrated land science, joining environmental, human, and remote sensing sciences, is emerging. It addresses questions about the impacts of land use and land cover changes, both on the environment and on the livelihoods of people [8]. This recently developed discipline led to the development of a large amount of case studies and data sets, with a corresponding plethora of methodologies for analysis. However, the complexity of causes, processes and impacts of land change has, so far, impeded the development of an integrated theory. Land science studies require versatile data analysis tools that can solve multitype pattern identification problems, ranging for instance from classification of satellite images into several land cover type classes in a map, to identifying land cover transition patterns in multi-temporal map data sets, or to prediction of pattern evolution through time.

Genetic Programming (GP) is the automated learning of computer programs, using Darwinian selection and Mendelian genetics as sources of inspiration [5]. In the last decade, GP has been extensively used both in Industry and Academia and it has produced a wide set of results that have been characterized as *human-competitive* [6]. Although in principle GP has the potential to evolve any kind of

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 334–343, 2013.
© Springer-Verlag Berlin Heidelberg 2013

solution, including decision trees, it has never been particularly suited for multiclass classification problems. This inadequacy, in many cases, derives not from limitations of the algorithm, but from the particular representation used for the solutions. The interested reader is referred to [13] for discussions on the difficulties of GP in facing multiclass classification problems and to [3] for a complete review of the state of the art methods in using GP for classification problems. It is a common procedure to address two-class classification problems using GP as regression ones, applying a cutoff to the predicted output. This approach can also be used for multiclass classification, but the approach in general becomes less effective as a larger number of classes is considered, and the performance of GP degrades to the point where other machine learning techniques become the only reasonable option. Therefore, even though GP has the potential to address the complexity of land science studies, the "simple" task of land use/land cover multiclass classification represents a potential obstacle.

Research in GP has recently focused on an aspect that was only marginally considered up to some years ago: the definition of methods based on the semantics of the solutions (see for instance [1,9]), where by semantics we generally mean the behavior of a program once it is executed on a set of data or, more specifically, the set of outputs a program produces on the training data. Using this definition of semantics (which is also the one that we adopt here), Moraglio *et al.* have recently defined new genetic operators, called geometric semantic genetic operators [10]. They have a number of theoretical advantages compared to the ones of standard GP; in particular, as proven in [10], they induce a unimodal fitness landscape on any problem consisting in finding the match between a set of input data and a set of known outputs (like for instance classification and regression). This should facilitate evolvability [4], making these problems potentially easier to solve for GP. However, they have a major drawback that makes them unusable in practice: they always create offspring that are larger than their parents, causing an exponential growth of the code in the population. We have proposed a new and very efficient implementation of the geometric semantic operators [12]. This new GP system evolves the semantics of the individuals without explicitly building their syntax, freeing us from dealing with exponentially growing trees and thus allowing us to test, for the first time, the potentiality of the semantic operators on complex real-life problems.

In this paper we want to assess how much improvement the geometric semantic operators introduce when compared to standard GP operators, in particular when dealing with multiclass classification problems in a 'regression and cutoff' manner. We tackle three problems: two real-life land cover/land use applications of four and ten classes, and a well-known benchmark of three classes.

The paper is organized as follows: Section 2 describes the geometric semantic operators of Moraglio *et al.* used in this work. Section 3 presents the experimental study, describing the test problems and settings, and discussing the results obtained. Section 4 concludes and describes our intended future work.

2 Geometric Semantic Operators

Many semantically aware methods presented so far [1,9] are indirect: search operators act on the syntax of the parents to produce offspring that are only accepted if some semantic criterion is satisfied. As reported by Moraglio *et al.* [10], this has at least two drawbacks: (i) these implementations are very wasteful as heavily based on trial-and-error; (ii) they do not provide insights on how syntactic and semantic searches relate to each other. To overcome these drawbacks, Moraglio *et al.* introduced new operators that directly search the semantic space.

To explain the idea, we first provide an example using Genetic Algorithms (GAs). Let us consider a GA problem in which the target solution is known and the fitness of each individual corresponds to its distance to the target (our reasoning holds for any distance measure used). This problem is characterized by a very good evolvability and it is in general easy to solve for GAs. In fact, for instance, if we use point mutation, any possible individual different from the global optimum has at least one neighbor (individual resulting from its mutation) that is closer to the target than itself, and thus is fitter. So, there are no local optima. In other words, the fitness landscape is unimodal. Similar considerations hold for box mutation and for many types of crossover, including various kinds of geometric crossover [7].

Now, let us consider the typical GP problem of finding a function that maps sets of input data into known target outputs (regression and classification are particular cases). The fitness of an individual for this problem is typically a distance between its predicted output values and the expected ones (error measure). Now let us assume that we are able to find a transformation on the syntax of an individual whose effect is just a random perturbation of one of its predicted output values. In other words, let us assume that we are able to transform an individual G into an individual H whose output values are like the outputs of G, except for one value, that is randomly perturbed. Under this hypothesis, we are able to map the considered GP problem into the GA problem discussed above, in which point mutation is used. So, this transformation, if known, would induce a unimodal fitness landscape on every problem like the considered one (e.g. regressions and classifications), allowing GP to have a good evolvability, at least on training data. The same also holds for transformations that correspond to box mutation or semantic crossovers. Although not without limitations, the work of Moraglio *et al.* [10] accomplishes this task, defining the following operators.

Definition 1. (Geometric Semantic Crossover). *Given two parent functions* $T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$*, the geometric semantic crossover returns the real function* $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$*, where* T_R *is a random real function whose output values range in the interval* $[0, 1]$.

The interested reader is referred to [10] for a formal proof of the fact that this operator corresponds to a geometric crossover on the semantic space, in the sense that it produces an offspring that stands between its parents in this space. We do not report the proof here, but we limit ourselves to remark that, even without a formal proof, we can have an intuition of it considering that the (only) offspring

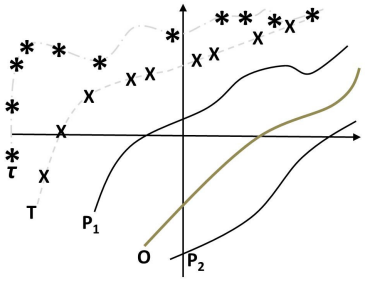


Fig. 1. Visual intuition of the fact that geometric semantic crossover creates an offspring that is at least not worse than the worst of its parents. In this toy example, offspring O (which stands between parents P_1 and P_2 in the semantic space by construction) is clearly closer to target T (training points represented by "×" symbols) than parent P_2. In Section 3 we also discuss the geometric properties of this operator on test data, represented by τ (test points represented by "∗" symbols).

generated by this crossover has a semantic vector that is a linear combination of the semantics of the parents with random coefficients included in $[0, 1]$ and whose sum is equal to 1. Moraglio *et al.* [10] also prove an interesting consequence of this fact: the fitness of the offspring cannot be worse than the fitness of the worst of its parents. Also in this case, we do not replicate the proof here, but we limit ourselves to providing a visual intuition of this property: in Figure 1 we represent a simple two-dimensional semantic space in which we draw a target function T (training points are represented by "×" symbols), two parents P_1 and P_2 and one of their offspring O (which by construction stands between its parents), plus a test set (composed by test points represented by "∗" symbols) that will be discussed in the final part of Section 3. It is immediately apparent from Figure 1 that O is closer to T than P_2 (which is the worst parent in this case). The generality of this property is proven in [10]. To constrain T_R in producing values in $[0, 1]$ we use the sigmoid function: $T_R = \frac{1}{1+e^{-T_{rand}}}$ where T_{rand} is a random tree with no constraints on the output values.

Definition 2. (Geometric Semantic Mutation). *Given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic mutation with mutation step ms returns the real function $T_M = T + ms \cdot (T_{R1} - T_{R2})$, where T_{R1} and T_{R2} are random real functions.*

Moraglio *et al.* [10] prove that this operator corresponds to a box mutation on the semantic space, and induces a unimodal fitness landscape. Even without a formal proof it is not difficult to have an intuition of it, considering that each element of the semantic vector of the offspring is a "weak" perturbation of the corresponding element in the parent's semantics. We informally define this perturbation as "weak" because it is given by a random expression centered in zero (the difference between two random trees). Nevertheless, by changing parameter ms, we are able to tune the "step" of this perturbation, and its importance.

3 Experimental Study

Test Problems. Two land cover/land use applications and a well-known benchmark have been used as test problems.

LANDMAP: Land cover mapping. The objective of this application is mapping land use/land cover types of Guinea-Bissau as function of six different metrics extracted from Landsat TM and ETM+ data for 2010. Mapping land use/land cover is one of the foremost requirements for planning, management and conservation of land and forest. This study considers 10 land cover types, among which three forest types (closed forest, open forest, savanna woodland) on a total of 6798 instances. Distinguishing between different forest types is a challenging task given the spectral similarity between them.

CASHEW: Cashew in West Africa. West Africa has one of the most modified tropical forest landscapes in the world, where tree cover is often part of a forest-savanna agriculture mosaic [11]. The objective of this application is to discriminate different land cover classes occurring in a forest and agriculture mosaic from 12 different metrics obtained from the RapidEye or Landsat Thematic Mapper data over Guinea-Bissau. The original data set contains 10 classes, on a total of 370 instances. However, as a preliminary study we used only four of these classes, on a total of 221 instances.

IRIS: Flower classification. This is a well-known benchmark problem available at the UCI Machine Learning Repository. The data set contains three classes of 50 instances each, where each class refers to a type of iris plant and each instance is described by four attributes.

Experimental Setting. We tackle each of the test problems with the two different GP systems: standard GP (ST-GP) and GP that uses the geometric semantic operators described in Section 2 (GS-GP). In all cases GP is used as if we were dealing with regression problems, i.e. the numeric class label is interpreted as the expected output value of the function to be learned.

For each of the GP systems, 50 independent runs have been performed with a population of 200 individuals. For each run, different randomly generated partitions of the data sets into training (70%) and test (30%) sets where used. The evolution stopped after 10000 fitness evaluations for both GP variants. Tree initialization was performed with the Ramped Half-and-Half method [5] with a maximum initial depth of 6. The function set contained the four arithmetic operators $+$, $-$, $*$, and / protected as in [5]. For each studied problem, the terminal set contained a number of variables equal to the number of features in the data set. Fitness was measured as the Root Mean Square Error (RMSE) between predicted and expected outputs, and tournament selection was used with tournament size of 4. The reproduction (replication) rate was 0.1, meaning that each selected parent has a 10% chance of being copied to the next generation instead of being engaged in breeding. ST-GP used standard subtree mutation and crossover (with uniform selection of crossover and mutation points among different tree levels), with probabilities 0.1 and 0.9 respectively. The new random

branch created for mutation has maximum depth 6. Selection for survival was elitist, guaranteeing the survival of the best individual from one generation to the next. No maximum tree depth was imposed. GS-GP used a higher mutation rate of 0.5, which was found to be necessary in order for GS-GP to properly explore the search space. The mutation step ms was 0.001.

Experimental Results. We compare the results of GS-GP and ST-GP obtained on training data and, in order to compare the generalization ability of the two methods, on out-of-sample test data. We report the RMSE on the training data, and the accuracy, expressed as the proportion of correctly classified samples, on the test data. In order to calculate accuracy, each predicted output is rounded to its nearest integer value, which represents the class label.

On Figure 2, the plots on the left report, for each studied problem, the evolution of the mean RMSE of the best individual on the training set over the 50 runs. They clearly show that GS-GP reaches the lowest RMSE on all the considered test problems. The boxplots on the right report the RMSE of the best individual on the training set at the end of each run. It can be observed that GS-GP produces solutions with a lower dispersion of RMSE than ST-GP on both LANDMAP and CASHEW problems. To analyze the statistical significance of these results, a set of tests has been performed on the RMSE values. As a first step, the Kolmogorov-Smirnov test has shown that the data are not normally distributed and hence a rank-based statistic has been used. More precisely, we have used the Mann-Whitney test [2], considering a confidence of 95% with a Bonferroni correction. According to this test, the results produced by GS-GP are statistically different from the ones produced by ST-GP on all the considered test problems; the respective p-values are reported in Table 1.

On Figure 3, the plots on the left report, for each studied problem, the evolution of the mean accuracy on the test set of the best individual on the training set over the 50 runs. Also in this case it is clear that GS-GP reaches higher accuracy, i.e. generalizes better, than ST-GP. The boxplots on the right report the accuracy on the test data of the best individual at the end of each run. Also here GS-GP produces solutions with a lower dispersion of RMSE than ST-GP. According to the Mann-Whitney test, the results produced by GS-GP are statistically different from the ones produced by ST-GP on all the considered test problems, as reported in Table 1.

Table 2 summarizes the results obtained on the different studied problems with both GP variants, in terms of minimum, maximum, median, mean and standard deviation of both RMSE on the training set and accuracy on the test set.

Discussion. From the above results we realize that neither ST-GP nor GS-GP overfit, since the accuracy values on the test set do not degrade during the evolution. However, GS-GP has obtained much better results than ST-GP in both training and test data. The good results on the training data were expected: the geometric semantic operators induce an unimodal fitness landscape, which facilitates evolvability. However, this could have caused a loss of generalization

Table 1. The p-values obtained comparing the median fitness (RMSE on the training set and accuracy on the test set) of GS-GP and ST-GP, using the Mann-Whitney statistical test.

	LANDMAP	**CASHEW**	**IRIS**
TRAINING	7.066e-018	1.914e-009	6.0178e-018
TEST	2.194e-017	4.778e-011	5.248e-018

Table 2. Summary of the results obtained on 50 independent runs. Training fitness is the RMSE (optimal fitness 0), while testing fitness is the classification accuracy (optimal fitness 1).

LANDMAP

	RMSE on TRAINING				
	Min	**Max**	**Median**	**Mean**	**Std Dev**
GS-GP	1.012	1.141	1.074	1.073	0.032
ST-GP	1.147	1.568	1.379	1.350	0.101
	ACCURACY on TEST				
	Min	**Max**	**Median**	**Mean**	**Std Dev**
GS-GP	0.612	0.743	0.696	0.690	0.033
ST-GP	0.337	0.645	0.518	0.511	0.082

CASHEW

	RMSE on TRAINING				
	Min	**Max**	**Median**	**Mean**	**Std Dev**
GS-GP	0.839	0.946	0.913	0.906	0.024
ST-GP	0.879	1.260	1.211	1.136	0.142
	ACCURACY on TEST				
	Min	**Max**	**Median**	**Mean**	**Std Dev**
GS-GP	0.239	0.388	0.313	0.317	0.034
ST-GP	0.194	0.328	0.224	0.243	0.045

IRIS

	RMSE on TRAINING				
	Min	**Max**	**Median**	**Mean**	**Std Dev**
GS-GP	0.096	0.176	0.143	0.145	0.016
ST-GP	0.380	0.494	0.445	0.444	0.030
	ACCURACY on TEST				
	Min	**Max**	**Median**	**Mean**	**Std Dev**
GS-GP	0.860	0.980	0.940	0.937	0.025
ST-GP	0.680	0.880	0.720	0.760	0.067

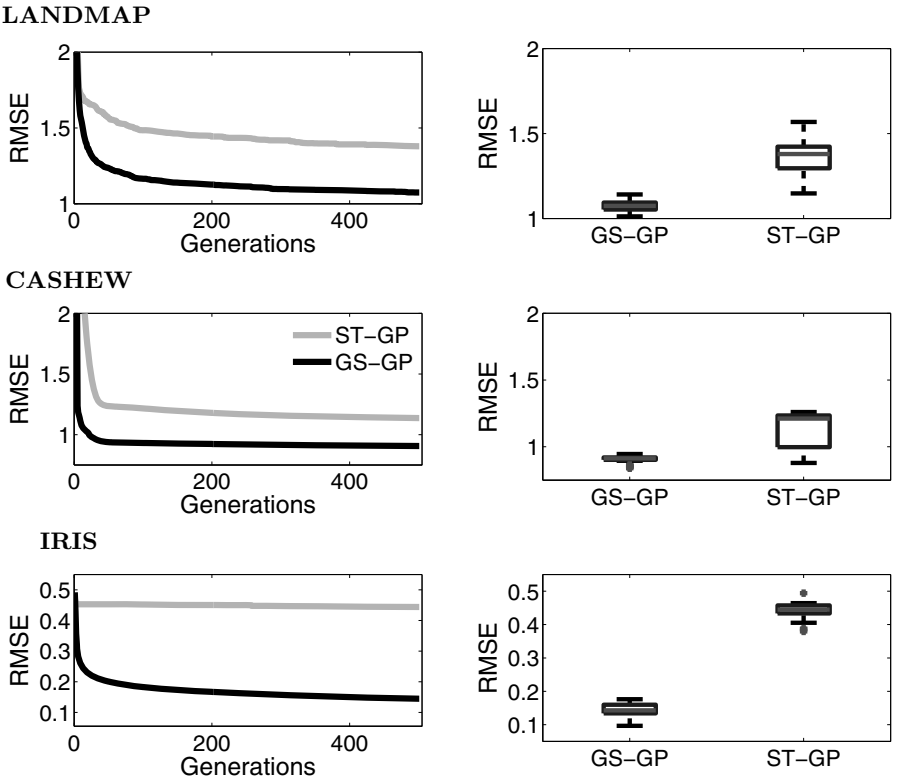


Fig. 2. Results obtained on the *training* set. Plots on the left: evolution of best fitness (RMSE), mean of 50 runs. Boxplots on the right: best fitness (RMSE) at the end of each run.

ability on the test data - it did not. Not so obvious at first sight, the geometric properties of the semantic operators hold *independently from the data* on which individuals are evaluated. In other words, geometric semantic crossover produces an offspring that stands between the parents also in the semantic space induced by test data. As a direct implication, following exactly the same argument as Moraglio *et al.* [10], each offspring is, in the worst case, not worse than the worst of its parents on the test set. This can be seen by looking back at Figure 1, where a simple test set τ is drawn (test points are represented by "$*$" symbols). Analogously, geometric semantic mutation produces an offspring that is a "weak" perturbation of its parent also in the semantic space induced by test data. This has an important consequence on the behavior of GS-GP on test data: even though the geometric semantic operators do not guarantee an improvement of test fitness each time they are applied (e.g. there is a very slight overfitting observed on the IRIS dataset with GS-GP, Figure 3), they at least guarantee that the possible worsening of the test fitness is "limited" (by the test fitness of the worst parent for crossover, and by the mutation step ms for mutation) [12].

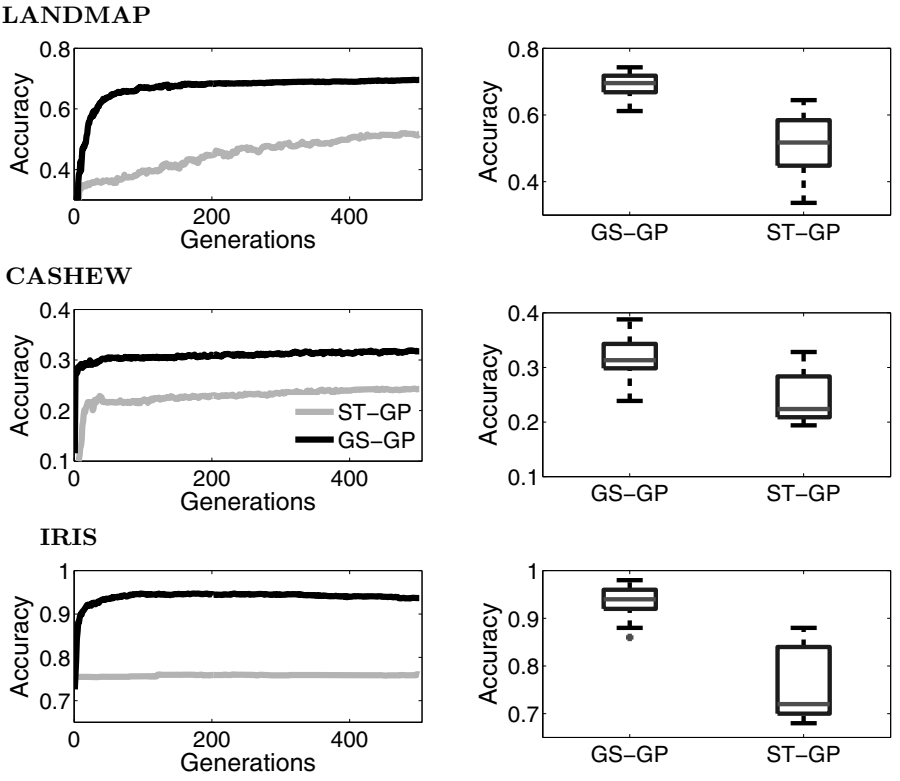


Fig. 3. Results obtained on the *test* set. Plots on the left: evolution of the accuracy on the test set of the best individual on the training set, mean of 50 runs. Boxplots on the right: accuracy on the test set at the end of each run.

4 Conclusions and Future Work

Multiclass classification is a common requirement of many land cover/land use applications, one of the pillars of land science studies. However, as reported in the literature, Genetic Programming (GP) is not particularly suited for this task. We have investigated the use of recently defined geometric semantic operators on multiclass classification problems, using two land cover/land use real-life applications, and one well-known benchmark, as test problems. Our results clearly indicate that GP that uses the geometric semantic operators (GS-GP) outperforms standard GP on all the studied problems. GS-GP returned much better results on training data without loss of generalization on test data. As future work we intend to qualitatively interpret the results achieved by GS-GP from the point of view of the applications, and compare its performance with other machine learning techniques using these and other multiclass classification problems, in order to determine if GS-GP is a competitive method for solving multiclass classification problems.

Acknowledgments. This work was partially supported by national funds through FCT under contract Pest-OE/EEI/LA0021/2011. The authors acknowledge projects "EnviGP - Improving Genetic Programming for the Environment and Other Applications" (PTDC/EIA-CCO/103363/2008), "Cashew in West Africa: socio-economic and environmental challenges of an expanding cash crop" (PTDC/AFR/117785/2010) and "MassGP - Improving Semantic Genetic Programming for Maritime Safety, Security and Environmental Protection" (PTDC/EEI-CTP/2975/2012) funded by the FCT, Portugal, and also project CarboVeg GB (funded by the Ministry of Environment, Portugal) and the Secretary of State of the Environment and Sustainable Development (SEAD, Guinea-Bissau).

References

1. Beadle, L., Johnson, C.G.: Semantic analysis of program initialisation in genetic programming. Genetic Programming and Evolvable Machines 10(3), 307–337 (2009)
2. Corder, G., Foreman, D.: Nonparametric statistics for Non-Statisticians. Wiley, New York (2009)
3. Espejo, P., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40(2), 121–144 (2010)
4. Gustafson, S., Vanneschi, L.: Crossover-based tree distance in genetic programming. IEEE Transactions on Evolutionary Computation 12(4), 506–524 (2008)
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
6. Koza, J.R.: Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines 11(3/4), 251–284 (2010); Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines
7. Krawiec, K., Lichocki, P.: Approximating geometric crossover in semantic space. In: GECCO 2009: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 987–994. ACM (2009)
8. Lambin, E.R., Geist, H., Rindfuss, R.R.: Local processes with global impacts. In: Lambin, E.R., Geist, H. (eds.) Land Use and Land Cover Change, pp. 1–8. Springer (2006)
9. McPhee, N.F., Ohs, B., Hutchison, T.: Semantic Building Blocks in Genetic Programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 134–145. Springer, Heidelberg (2008)
10. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric Semantic Genetic Programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
11. Norris, K., Asase, A., Collen, B., Gockowksi, J., Mason, J., Phalan, B., Wade, A.: Biodiversity in a forest-agriculture mosaic – the changing face of west african rainforests. Biological Conservation 143, 2341–2350 (2010)
12. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP applied to predicting pharmacokinetic parameters. In: Proceedings of EuroGP-2013. Springer (to appear, 2013)
13. Zhang, M., Smart, W.: Multiclass Object Classification Using Genetic Programming. 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. 369–378. Springer, Heidelberg (2004)

Adding Chaos to Differential Evolution for Range Image Registration

Ivanoe De Falco[1], Antonio Della Cioppa[2], Domenico Maisto[1], Umberto Scafuri[1], and Ernesto Tarantino[1]

[1] ICAR-CNR, Via P. Castellino 111, 80131 Naples, Italy
{ivanoe.defalco,domenico.maisto,umberto.scafuri, ernesto.tarantino}@na.icar.cnr.it
[2] Natural Computation Lab, DIEM, University of Salerno Via Ponte don Melillo 1, 84084 Fisciano (SA), Italy
adellacioppa@unisa.it

Abstract. This paper presents a method for automatically pair–wise registering range images. Registration is effected adding chaos to a Differential Evolution technique and by applying the Grid Closest Point algorithm to find the best possible transformation of the second image causing 3D reconstruction of the original object. Experimental results show the capability of the method in picking up efficient transformations of images with respect to the classical Differential Evolution. The proposed method offers a good solution to build complete 3D models of objects from 3D scan datasets.

1 Introduction

In computer vision Range Image Registration (RIR) is a fundamental task used for integrating information acquired under diverse viewing angles (multi–view analysis). During the years several multi–view RIR techniques have been developed [21, 22] to tackle many practical applications, such as 3D modeling ranging from medical imaging, remote sensing, physical objects, digital archaeology, restoration of historic buildings, virtual museum, artificial vision, reverse engineering and computer–aided design [19]. These applications require the construction of precise 3D models preserving as much information as possible.

Since a physical object cannot be completely scanned with a single image due to the occlusions and the limited field of view of a sensor, a set of range images taken from different positions are required to supply the information needed to construct the whole 3D model. These multiple images are acquired by a range scanner involved in surface reconstruction. The registration strategy can differ according to whether all range views of the objects are registered at the same time (simultaneous registration) or only a pair of adjacent range images are processed in every execution (pair–wise registration). This paper is focused on the pair–wise registration of range images.

The objective of the registration process for two views consists in finding the best spatial transformation that, when applied to one view, aligns it with the

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 344–353, 2013.
© Springer-Verlag Berlin Heidelberg 2013

other in a common coordinate system. Such a transformation estimation is usually formulated as an optimization problem solved by an iterative procedure. In literature there exist several methods for RIR approaches based on the Iterative Closest Point (ICP) algorithm [2] which requires a good prealignment of the views to converge to the global optimum. Unfortunately, an exhaustive exploration of the search space of all the candidate solutions is impracticable in case of a large number of degrees of freedom of the transformation, and thus stochastic optimization algorithms, such as Evolutionary Algorithms (EAs) [1], capable of providing a solution acceptably close to the global optimum in a reasonable time, have been successfully applied to complex real–world problems in computer vision and image registration [6, 8–10, 14, 22].

Differential Evolution (DE) [18] is an EA which has proven fast and reliable to face several multivariable optimization tasks in many areas [7, 18]. Here we propose and examine the ability of adaptive updating schemes for DE based on chaos theory to perform automatic pair–wise image registration by exploiting the Grid Closest Point (GCP) [25] transformation with no *a priori* knowledge of the pose of the views. This system is tested on a set of eight 3D range images.

The paper structure is as follows: Section 2 describes the state of the art. Section 3 contains the description of a DE adopting chaotic sequences and illustrates the application of our system to the registration task defining the encoding and the fitness of the related optimization problem. Section 4 reports the results achieved by our tool. The last section contains final remarks and future works.

2 State of the Art

Several surveys on RIR are available in literature. For example, ICP methods centered on the point–to–point and point–to–plane correspondences are reported in [20]. A complete study focused on pair–wise registration is presented in [5], while different techniques for both pair–wise and multi–view registration, and a new classification, can be found in [21].

In the last two decades, EAs have been extensively applied to the image registration problems. Differently from methods based on the ICP algorithm, the most popular family of methods to date, EAs need neither rough nor near–optimal prealignment of the images to proceed. An extensive review of evolutionary image registration methods is reported in [22].

He and Narayana [13] propose a real coding scheme that makes use of arithmetic crossover and uniform mutation operators within an elitist generational model including a restart mechanism. The evolutionary method uses a real–coded Genetic Algorithm (GA) to estimate the rigid transformation and a local search procedure to refine the obtained preliminary solution.

Chow et al. [3] still propose the use of real–coded GA considering a rigid transformation but introduce a crossover operator that randomly selects the number of genes to be swapped and a different sophisticated restart mechanism.

Silva et al. [24] face the pair–wise registration problem of range images captured by a 3D laser range scanner through a parameter–based approach for rigid

transformations. The proposed technique is inspired to the steady-state GA used together with a hill–climbing algorithm to improve the precision of the results.

In [14] a new method for pair–wise registration is introduced. The novelty consists in the inclusion in the solution vector of a surface overlap parameter and the use of the trimmed square metric as objective function.

Cordón et al. [4] present a Scatter Search EA adopting a matching-based approach while Santamaria et al. propose different memetic–based image registration techniques to deal with 3D reconstruction of forensic objects [23].

3 Chaotic Differential Evolution

The performance of DE is sensitive to the choice of the scale factor F and of the crossover rate CR [15]. In this paper we introduce several adaptive updating schemes for setting such parameters based on chaos theory [26]. Chaos describes the complex behavior of a nonlinear deterministic system which is dynamic, pseudo–random, ergodic, and sensitive to initial conditions [17]. One of the simplest and most commonly used dynamic systems evidencing chaotic behavior is the logistic map described by the following quadratic recurrence equation:

$$y_{t+1} = \mu \cdot y_t \cdot (1 - y_t), \text{ with } t = 1, 2, 3, \ldots \tag{1}$$

where μ is a positive constant sometimes known as *biotic potential*. The behavior of the system is greatly affected by the value of μ which determines whether y stabilizes at a constant size, oscillates among a limited sequence of sizes, or behaves chaotically in an unpredictable pattern. A very small difference in the initial value y_1 could cause large differences in its long–time behavior. Eq. (1) exhibits chaotic dynamics with values within the range $[0, 1]$ at μ approximately 3.57. Beyond $\mu = 4$ and $y_1 \neq 0, 0.25, 0.50, 0.75, 1$, the values eventually leave the interval $[0, 1]$ and diverge for almost all initial values.

The application of chaotic sequences can be a viable means to improve the exploration capability of an optimization algorithm. In fact, due to the ergodicity property, chaos can be used to enrich the searching behavior and to avoid being trapped into local optima [26]. Our schemes, based on logistic maps, are implemented in a global and a local strategy. The control parameters are randomly initialized in both strategies.

In the *global strategy* (A–GlChDE) the same values for F and CR are used for all the individuals $\mathbf{x}_i$ (potential solutions of the problem), and the control parameters between consecutive generations t and $t + 1$ are updated as follows:

$$\begin{cases} F_{t+1} & = \mu \cdot F_t \cdot (1 - F_t) \\ CR_{t+1} & = \mu \cdot CR_t \cdot (1 - CR_t) \end{cases} \tag{2}$$

In the *local strategy* (A–LocChDE) F and CR are associated to each individual $\mathbf{x}_i$ in the population, and adjusted according to independent logistic chaotic sequences. In formulae:

$$\begin{cases} F_{t+1}(\mathbf{x}_i) & = \mu \cdot F_t(\mathbf{x}_i) \cdot (1 - F_t(\mathbf{x}_i)) \\ CR_{t+1}(\mathbf{x}_i) & = \mu \cdot CR_t(\mathbf{x}_i) \cdot (1 - CR_t(\mathbf{x}_i)) \end{cases} \tag{3}$$

Moreover, two further schemes adopt a form of control of the quality of the current parameters based on fitness evaluations. This allows further exploiting the most promising parameter couples and, at the same time, assuring the exploration of new potentially propitious search parameter values.

Let us suppose that $\Phi(\cdot)$ is the fitness function for a minimization problem. For the global strategy F and CR at the time $t+1$ are updated by using Eq. (2) only if $\langle \Phi_t \rangle \geqslant \langle \Phi_{t-1} \rangle$, i.e., the average fitness in the population at the generation t is not better than that at the previous generation $t-1$. This strategy is referred to as A–GlChDE$_{Avg}$.

For the local strategy $F_{t+1}(\mathbf{x}_i)$ and $CR_{t+1}(\mathbf{x}_i)$ of each individual $\mathbf{x}_i$ at the generation $t+1$ are updated as follows: if $\Phi_t(\mathbf{x}_i) \geqslant \langle \Phi_{t-1} \rangle$, i.e., the fitness of the current individual is not better than the average fitness value in the population at the previous generation, the values of its control parameters are updated based on independent logistic chaotic sequences by means of Eq. (3). On the contrary, the current values of the control parameters are retained. Such a strategy is indicated as A–LocChDE$_{Avg}$.

In these last two strategies the number of parameter couples involved changes dynamically over generations depending on the fitness feedback from the search.

It should be noted that the introduction of the above strategies does not impact computational complexity of the whole DE algorithm. In fact, in the case of the computationally heaviest strategy, i.e., A–LocChDE$_{Avg}$, the computational complexity of the added scheme is $O(t_{\max} p)$, where p is the population size and $t_{\max}$ is the maximum number of generations.

3.1 Encoding and Fitness

Given two input images, named scene $I_s = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ and model $I_m = \{\boldsymbol{p}'_1, \ldots, \boldsymbol{p}'_m\}$ with n and m points respectively, RIR aims to find the best possible Euclidean motion f for I_s determined by the rotation $R = (\theta, Ax_x, Ax_y, Ax_z)$ and the translation $\boldsymbol{t} = (t_x, t_y, t_z)$ with θ being the angle and Ax the axis of rotation. Then the transformed points are denoted as: $f(\boldsymbol{p}_i) = R(\boldsymbol{p}_i) + \boldsymbol{t}, \;\; i = 1, \ldots, n$. Actually unit quaternions are used to manage rotations in order to avoid singularities and discontinuities, e.g. gimbal lock.

The pair–wise RIR problem can then be seen as a numerical optimization problem in which solutions are encoded as seven–dimensional vectors of real values $\boldsymbol{x} = (\theta, Ax_x, Ax_y, Ax_z, t_x, t_y, t_z)$. The aim is to search the Euclidean transformation f^* achieving the best alignment of both $f(I_s)$ and I_m based on the chosen similarity metric Φ to optimize:

$$f^* = \arg\min_f \Phi(I_s, I_m; f) \tag{4}$$

Due to its robustness in presence of outliers (i. e., acquired noisy range images), the similarity metric Φ usually considered in 3D modeling is the median square error (*MedSE*) [19]. It can be formulated as:

$$\Phi(I_s, I_m; f) = MedSE(d_i^2), \;\; \forall i = \{1, \ldots, n\} \tag{5}$$

Table 1. The results

Algorithm		Angel	Bird	Buddha	Bunny	Duck	Frog	Lobster	Teletubby
DE	Φ_b	**1.620**	1.360	1.832	**1.182**	1.252	1.299	**1.451**	**1.325**
	$\langle\Phi\rangle$	1.622	1.878	1.841	1.183	1.285	1.362	1.458	**1.325**
	σ_Φ	0.002	0.371	**0.010**	0.005	0.033	0.082	0.011	0.003
A–LocChDE	Φ_b	**1.620**	**1.298**	**1.540**	**1.182**	**1.195**	**1.251**	**1.451**	**1.325**
	$\langle\Phi\rangle$	**1.620**	1.468	**1.817**	**1.182**	**1.211**	**1.293**	**1.451**	**1.325**
	σ_Φ	**0.000**	0.410	0.085	**0.000**	**0.020**	**0.016**	**0.000**	**0.000**
A–LocChDE$_{Avg}$	Φ_b	**1.620**	**1.298**	1.832	**1.182**	**1.195**	**1.251**	**1.451**	**1.325**
	$\langle\Phi\rangle$	**1.620**	**1.418**	1.865	1.192	1.232	1.334	**1.451**	**1.325**
	σ_Φ	0.001	**0.326**	0.026	0.047	0.048	0.120	**0.000**	**0.000**
A–GlChDE	Φ_b	**1.620**	**1.298**	1.661	**1.182**	**1.195**	**1.251**	**1.451**	**1.325**
	$\langle\Phi\rangle$	**1.620**	1.643	1.835	**1.182**	1.221	1.327	**1.451**	**1.325**
	σ_Φ	**0.000**	0.475	0.018	**0.000**	0.039	0.118	**0.000**	**0.000**
A–GlChDE$_{Avg}$	Φ_b	**1.620**	**1.298**	1.661	**1.182**	**1.195**	**1.251**	**1.451**	**1.325**
	$\langle\Phi\rangle$	**1.620**	1.531	1.846	**1.182**	1.239	1.363	**1.451**	1.358
	σ_Φ	0.002	0.484	0.047	**0.000**	0.058	0.160	**0.000**	0.130

where *MedSE* corresponds to the median value of all the squared Euclidean distances, $d_i^2 = ||f(\boldsymbol{p}_i) - \boldsymbol{p}'_j||^2$ $(j = 1, \ldots, m)$, between the transformed scene point, $f(\boldsymbol{p}_i)$, and its corresponding closest point, $\boldsymbol{p}'_j$, in the model view I_m. To speed up the computation of the closest point the GCP transform is used [25].

4 Experimental Results

To investigate the behavior of the presented chaotic DE algorithms in the RIR domain, a set of benchmarks from the image repository [16] collected at Signal Analysis and Machine Perception Laboratory (SAMPL) at the Ohio State University has been taken into account. From that repository, the following eight objects have been considered: Angel, Bird, Buddha, Bunny, Duck, Frog, Lobster, and Teletubby. For each of them the couple of images taken at angles 0 and 40 degrees have been chosen as exemplary instances for pair–wise RIR.

Throughout our experiments we have used a DE/*rand*/1/*bin*. Its parameter setting has been arbitrarily chosen as follows: $p = 30$, $t_{\max} = 500$, $CR = 0.3$ and $F = 0.7$. For the logistic map, we have used $\mu = 4$ and $y(1) =]0, 0.5[- \{0.25\}$, where 0.25 should be avoided because $y(t)$ goes to a fixed point.

For each algorithm and for each problem Tab. 1 reports the best final value Φ_b achieved in 25 runs, the average value $\langle\Phi\rangle$ over the 25 final values, and the related standard deviation σ_Φ. For each such index, the table shows in bold the algorithm with the best value for each problem. As a first remark, the results achieved by DE are improved by those of all the chaos–based algorithms. This evidences that our idea of adding chaos to the classical DE is sensible.

Among the chaotic algorithms, A–LocChDE always achieves the best value for Φ_b and on seven problems obtains the best performance for $\langle\Phi\rangle$, so it appears to be the best performing chaotic version. Moreover, on six out of the eight problems it shows the lowest value for σ_Φ, meaning that it is very robust too

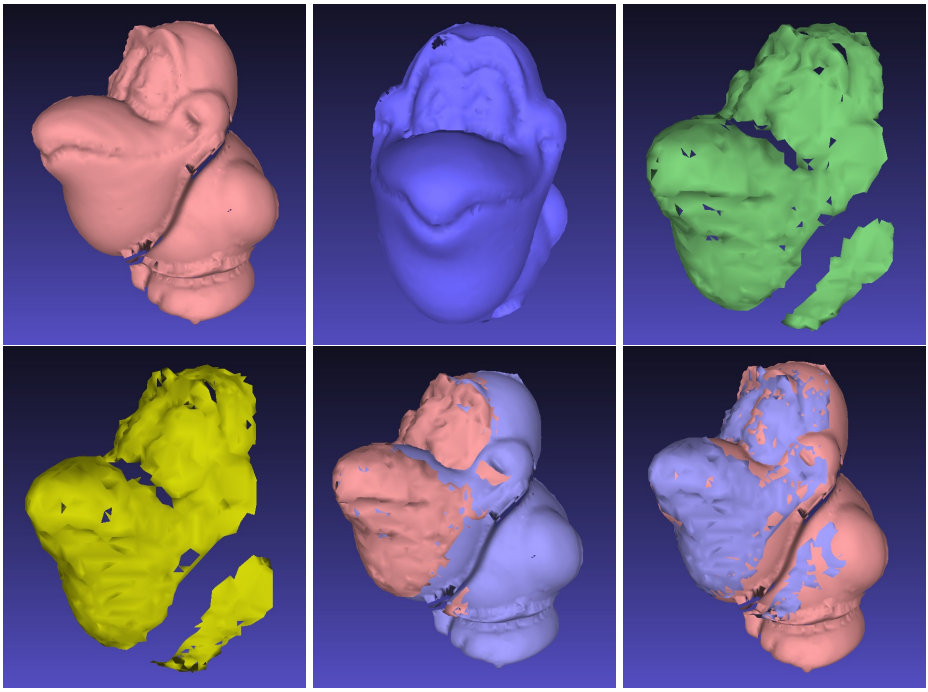


Fig. 1. Some examples of results for Bird problem. Top left: original image at zero degrees. Top center: original image at forty degrees. Top right: the best transformation of the second image achieved by DE. Bottom left: the best transformation of the second image achieved by A–LocChDE. Bottom center: Bird reconstructed by DE. Bottom right: Bird reconstructed by A–LocChDE.

independently of the starting seed. The remaining three chaotic algorithms seem to be about equivalent one another in terms of performance.

As an example of the results obtained by the different algorithms, Fig. 1 shows the outcome for Bird object. A–LocChDE causes many more points of the second image to be correctly transformed and contribute to successfully reconstruct the object surface than DE does: this takes place in the areas of the left sides of the left eye, of the face between mouth and ear, and of the body.

The improvement in the results achieved when adding chaos–based evolution to DE can be visually appreciated in the two images shown in Fig. 2.

In the left pane of the figure, dealing with classical DE, it can be seen that the cloud of points representing the second image after the transformation, sketched in light blue, is in several parts of the image quite far from that representing the first image, drawn in white. This is true especially in the area in front of the nose and of the mouth. The right pane shows the same issue for the best transformation proposed by A–LocChDE. The difference between corresponding pairs of points is in this case much lower, as the same area investigated in the

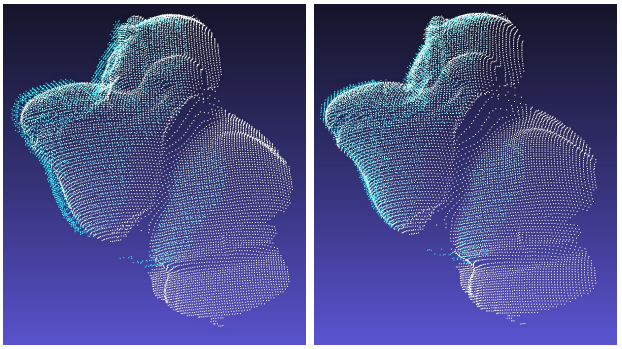


Fig. 2. The differences between the cloud of points representing the first image (in white) and that for the best transformation of the second image (in light blue). Left pane: classical DE. Right pane: A–LocChDE.

other pane evidences. This is just an example, yet holds true for all the faced problems. Therefore, the conclusion can be drawn that adding chaos to DE allows achieving transformations of the second image that are closer to the first one, thus leading to better 3-D object reconstructions.

4.1 Statistical Analysis

To compare the algorithms from a statistical point of view, a classical approach based on nonparametric statistical tests has been carried out, following [11, 12]. To do so, the ControlTest package [12] has been used. It is a Java package freely downloadable at *http://sci2s.ugr.es/sicidm/*, developed to compute the rankings for these tests, and to carry out the related post–hoc procedures and the computation of the adjusted p–values.

The results for the one–to–all analysis are reported in the following. Table 2 contains the results of the Friedman, Aligned Friedman, and Quade tests in terms of average rankings obtained by all the DE versions. The last two rows show the statistic and the p–value for each test, respectively. For Friedman and Aligned Friedman tests the statistic is distributed according to chi–square with 4 degrees of freedom, whereas for Quade test it is distributed according to F–distribution with 4 and 28 degrees of freedom. In each of the three tests, the lower the value for an algorithm, the better the algorithm. A–LocChDE turns out to be the best in all of the three tests. Among the other algorithms, their order is in all the tests the following: A–GlChDE is always the second best heuristic, A–LocChDE$_{Avg}$ is the third, followed by A–GlChDE$_{Avg}$, and finally the classical DE is the fifth.

Furthermore, with the aim to examine if some hypotheses of equivalence between the best performing algorithm and the other ones can be rejected, the complete statistical analysis based on the post–hoc procedures ideated by Holm, Hochberg, Hommel, Holland, Rom, Finner, and Li has been carried out following [12]. Moreover, the adjusted p–values have been computed by means of [12].

Table 2. Average rankings of the algorithms

Algorithm	Friedman	Aligned Friedman	Quade
DE	4.188	29.563	4.194
A–LocChDE	1.813	11.188	1.597
A–LocChDE$_{Avg}$	3.063	19.563	2.903
A–GlChDE	2.438	17.438	2.542
A–GlChDE$_{Avg}$	3.500	24.750	3.764
test statistic	10.850	6.566	3.500
p–value	0.028	0.161	0.019

Table 3. Results of post–hoc procedures for Friedman(top), Aligned Friedman (center), and Quade (bottom) tests over all tools (at $\alpha = 0.05$)

i	Algorithm	$z = (R_0 - R_i)/SE$	p	Holm/Hochberg/Hommel	Holland	Rom	Finner	Li
4	DE	3.004	0.003	0.013	0.013	0.013	0.013	0.030
3	A–GlChDE$_{Avg}$	2.135	0.033	0.017	0.017	0.017	0.025	0.030
2	A–LocChDE$_{Avg}$	1.581	0.114	0.025	0.025	0.025	0.038	0.030
1	A–GlChDE	0.791	0.429	0.050	0.050	0.050	0.050	0.050
Th				0.017/0.013/0.017	0.017	0.013	0.025	0.030

i	Algorithm	$z = (R_0 - R_i)/SE$	p	Holm/Hochberg/Hommel	Holland	Rom	Finner	Li
4	DE	3.14	0.002	0.013	0.013	0.013	0.013	0.038
3	A–GlChDE$_{Avg}$	2.320	0.020	0.017	0.017	0.017	0.025	0.038
2	A–LocChDE$_{Avg}$	1.433	0.152	0.025	0.025	0.025	0.038	0.038
1	A–GlChDE	1.069	0.285	0.050	0.050	0.050	0.050	0.050
Th				0.017/0.013/0.017	0.017	0.013	0.038	0.038

i	Algorithm	$z = (R_0 - R_i)/SE$	p	Holm/Hochberg/Hommel	Holland	Rom	Finner	Li
4	DE	2.070	0.038	0.013	0.013	0.013	0.013	0.029
3	A–GlChDE$_{Avg}$	1.727	0.084	0.017	0.017	0.017	0.025	0.029
2	A–LocChDE$_{Avg}$	1.041	0.299	0.025	0.025	0.025	0.038	0.029
1	A–GlChDE	0.753	0.452	0.050	0.050	0.050	0.050	0.050
Th				0.013/—/0.013	0.013	—	0.013	0.029

Tables 3 reports the results of this analysis performed at a level of significance $\alpha = 0.05$. In this table the other algorithms are ranked in terms of distance from the best performing one, and each algorithm is compared against this latter to investigate whether or not the equivalence hypothesis can be rejected. For each algorithm each table reports the z value, the unadjusted p–value, and the adjusted p–values according to the different post-hoc procedures. The variable z represents the test statistic for comparing the algorithms, and its definition depends on the main nonparametric test used. In [12] all the different definitions for z, corresponding to the different tests, are reported. The last row in each sub–table contains for each procedure the threshold value Th such that the procedure considered rejects those equivalence hypotheses that have an adjusted p–value lower than or equal to Th.

Summarizing the results of these tables, A–LocChDE is for all the three tests and for all the post–hoc procedures statistically better than the classical DE. Therefore, statistical analysis confirms that the introduction of chaos into the classical DE improves the ability of this latter to efficiently face RIR problem.

5 Conclusions and Future Works

In this paper an adaptive chaotic Differential Evolution technique has been investigated to optimize a 3D rigid transformation for automatic pair–wise registration of range images without considering any previous knowledge of the pose of the view. The Grid Closest Point transformation has been used to speed up the computation of the closest points. The experimental phase, carried out on a set of benchmark range images, shows that our chaotic evolutionary system performs better than the classical DE and is promising, yet there is plenty of work still to do to further evaluate the effectiveness of our system and its limitations.

Firstly, we plan to compare our method against other chaotic evolutionary systems. Moreover, the use of other chaotic maps will be investigated. Lastly, we aim to explore the behavior of the proposed algorithms in dealing with affine transformations.

References

1. Bäck, T., Fogel, D., Michalewicz, Z.: Handbook of Evolutionary Computation. IOP Publishing Ltd. (1997)
2. Besl, P.J., McKay, N.D.: A method for registration of 3–d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2), 239–256 (1992)
3. Chow, K.W., Tsui, T.: Surface registration using a dynamic genetic algorithm. Pattern Recognition 37, 105–117 (2004)
4. Cordón, O., Damas, S., Santamaría, J., Martí, R.: Scatter search for the point-matching problem in 3D image registration. Informs J. on Computing 20(1), 55–68 (2008)
5. Dalley, G., Flynn, P.: Pair-wise range image registration: a study in outlier classification. Computer Vision and Image Understanding 87(1-3), 104–115 (2002)
6. Damas, S., Cordón, O., Santamaría, J.: Medical image registration using evolutionary computation: An experimental survey. IEEE Computational Intelligence Magazine 6(4), 26–42 (2011)
7. De Falco, I., Della Cioppa, A., Iazzetta, A., Tarantino, E.: An evolutionary approach for automatically extracting intelligible classification rules. Knowledge and Information Systems 7(2), 179–201 (2005)
8. De Falco, I., Della Cioppa, A., Maisto, D., Scafuri, U., Tarantino, E.: Satellite Image Registration by Distributed Differential Evolution. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 251–260. Springer, Heidelberg (2007)
9. De Falco, I., Della Cioppa, A., Maisto, D., Tarantino, E.: Differential evolution as a viable tool for satellite image registration. Applied Soft Computing 8(4), 1453–1462 (2008)
10. De Falco, I., Maisto, D., Scafuri, U., Tarantino, E., Della Cioppa, A.: Distributed differential evolution for the registration of remotely sensed images. In: 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, pp. 358–362. IEEE (2007)
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)

12. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1, 3–18 (2011)
13. He, R., Narayana, P.A.: Global optimization of mutual information: application to three–dimensional retrospective registration of magnetic resonance images. Comput. Med. Imag. Grap. 26, 277–292 (2002)
14. Lomonosov, E., Chetverikov, D., Ekárt, A.: Pre-registration of arbitrarily oriented 3D surfaces using a genetic algorithm. Pattern Recognition Lett. 27(11), 1201–1208 (2006)
15. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33(1), 61–106 (2010)
16. Ohio State University: SAMPL image repository (2009), http://sampl.eng.ohio-state.edu/~sampl/database.htm
17. Peitgen, H.O., Jürgens, H., Saupe, D.: Chaos and Fractals: New Frontiers of Science. Springer (2004)
18. Price, K., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series. Springer (2005)
19. Rodrigues, M., Fisher, R., Liu, Y.: Introduction: Special issue on registration and fusion of range images. Computer Vision and Image Understanding 87, 1–7 (2002)
20. Rusinkiewicz, S., Levoy, M.: Efficient variant of the icp algorithm. In: Third International Conference on 3–D Digital Imaging and Modeling, pp. 145–152 (2001)
21. Salvi, J., Matabosch, C., Fofi, D., Forest, J.: A review of recent range image registration methods with accuracy evaluation. Image and Vision Computing 25(5), 578–596 (2007)
22. Santamaría, J., Cordón, O., Damas, S.: A comparative study of state–of–the–art evolutionary image registration methods for 3d modeling. Computer Vision and Image Understanding 115(9), 1340–1354 (2011)
23. Santamaría, J., Cordón, O., Damas, S., Aleman, I., Botella, M.: A scatter search-based technique for pair-wise 3d range image registration in forensic anthropology. Soft Computing-A Fusion of Foundations, Methodologies and Applications 11(9), 819–828 (2007)
24. Silva, L., Bellon, O.R.P., Boyer, K.L.: Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. IEEE Transactions on Pattern Analysis 27(5), 762–776 (2005)
25. Yamany, S.M., Ahmed, M.N., Heyamed, E.E., Farag, A.A.: Novel surface registration using the grid closest point (cgp) transform. In: Int. Conf. on Image Processing, vol. 3, pp. 809–813. IEEE Press (1998)
26. Yu, G., Wang, X., Li, P.: Application of chaotic theory in differential evolution algorithms. In: Sixth International Conference on Natural Computation, vol. 7, pp. 3816–3820. IEEE Press (2010)

Genetic Programming for Automatic Construction of Variant Features in Edge Detection

Wenlong Fu[1], Mark Johnston[1], and Mengjie Zhang[2]

[1] School of Mathematics, Statistics and Operations Research
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
[2] School of Engineering and Computer Science
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
wenlong.fu@msor.vuw.ac.nz, {mark.johnston,mengjie.zhang}@vuw.ac.nz

Abstract. Basic features for edge detection, such as derivatives, can be further manipulated to improve detection performance. However, how to effectively combine different basic features remains an open issue and needs to be investigated. In this study, Genetic Programming (GP) is used to automatically and effectively construct rotation variant features based on basic features from derivatives, F-test, and histograms of images. To reduce computational cost in the training stage, the basic features only use the horizontal responses to construct new horizontal features. These new features are then combined with their own rotated versions in the vertical direction in the testing stage. The experimental results show that the rotation variant features constructed by GP combine advantages from the basic features, reduce drawbacks from basic features alone, and improve the detection performance.

Keywords: Genetic Programming, Edge Detection, Feature Construction.

1 Introduction

Features in edge detection are functions of raw pixel values in an image relative to a local point and are used in the process of classifying pixels as edge points or not. Features can be categorised as variant features and invariant features. An invariant feature is not affected by image rotation, however, a variant feature has different responses on pixels from an image when the image is rotated. Therefore, an invariant feature method usually uses one vector to store extracted results, but a variant feature method needs multiple vectors based on different angles to store extracted results [2,12].

Since edge detection is subjective, various approaches have been developed to extract features for detecting edges [2,12,15]. In general, one feature for edge detection is not sufficient to fully identify the edges in an image. For instance, features based on image gradients are not good to detect texture edges [12]. A set of features can be combined to improve detection performance [12,14,15].

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 354–364, 2013.
© Springer-Verlag Berlin Heidelberg 2013

However, e.g., a Boosted Edge Learning algorithm [2] using approximately 50000 features for natural images only has similar evaluation performance to a contour detector proposed in [12] with nine local features. Features combined by different learning approaches might have very similar detection performance [12]. Therefore, how to efficiently and effectively combine features still needs to be investigated.

Genetic Programming (GP) has been utilised to evolve low-level edge detectors [6,16]. The existing work is mainly concentrated on constructing low-level edge detectors via choosing raw pixels [19], or combining image operators [10]. These works show that GP can evolve good edge detectors [6,10,19], and it is promising to use GP for automatically constructing features for edge detection based on existing feature extraction methods.

The overall goal of this paper is to investigate using GP to automatically constructing rotation variant features for edge detection from a set of basic variant features (only extracted based on a fixed direction). In the same extracted feature, the distribution of the responses to one direction should be similar to the distribution of the responses to other directions. Note that we do not consider pixel positions in an extracted feature based on one direction and there are enough different directional edges. Therefore, if we can find a method to get responses on one direction, it might be possible to use the method to obtain responses on other directions. During the training stage, we only need partial information (from one direction) of the training data to construct new features so that the evaluation cost on these new features is reduced.

The image derivatives and histogram derivatives [12] are popularly used to train contour detectors, so they are used as basic features. In order to enrich the set of features for edge detection, an F-test is also used as a basic feature. Based on these three basic features (only extracted from the horizontal direction), composite features (in the horizontal direction) will be constructed automatically by GP. Specifically, we would like to investigate the following research objectives: (1) whether the features constructed by GP only based on a fixed direction can perform edge detection via combining with their own rotated versions; (2) whether the features constructed by GP can improve the detection performance, compared with each basic feature alone; (3) whether the constructed features are better than the features constructed by a popular method, such as a Bayesian model.

In the remainder of the paper, Section 2 briefly describes some relevant background. Section 3 presents how GP can be used to construct variant features for edge detection. After presenting the experimental design in Section 4, Section 5 describes the results with discussions. Section 6 gives conclusions and future work directions.

2 Background

2.1 Edge Detection

Fig. 1 shows a general edge detection process flow. For an image I, an intermediate result I' will be obtained after pre-processing. The feature extraction

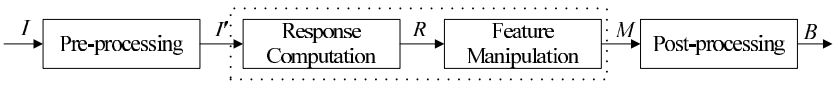


Fig. 1. General edge detection flow

stage is divided into two phases, namely response computation and feature manipulation. In the response phase, the computation can come from gradients, and also statistics [11], and a set of features R is obtained. Note that, some edge detectors combine the pre-processing and response computation together, such as the image gradients after filtering noise by a Gaussian filter in the Canny detector [1]. In the feature manipulation phase, feature selection [2] and further feature construction [8,12] are included, and the output is a set of features M. After post-processing, a final edge map B is obtained. In general, the post-processing techniques can usefully follow most feature extraction approaches. The evaluation of the features in edge detection is important [13] although the performance evaluation usually focuses on final edge maps.

Feature extraction mainly works on local features for the sake of simplicity and ease of implementation. Local features mainly come from gradient computation, such as image derivatives [1,8] or local histogram derivatives [12]. The image derivatives based on different directions have been widely applied in different edge detectors [15], such as the popular Canny edge detector [1]. The derivative for a Gaussian filter extracted in the horizontal direction (horizontal edges, not horizontal derivatives) are described in Equation (1), where (u, v) is the position of a point relative to a center pixel, and σ is the parameter. After performing convolution ($\circledast$) between image $I(x, y)$ with the derivative $g_{0^\circ}(u, v)$, the image Gaussian filter derivatives $T_{gd,0^\circ}$ are obtained (see Equation (2)).

$$g_{0^\circ}(u, v) = -\frac{v}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \tag{1}$$

$$T_{gd,0^\circ}(x, y) = g_{0^\circ}(u, v) \circledast I(x, y) \tag{2}$$

Local image histogram derivatives have shown good performance for detecting edges [12]. The image local histogram gradients are extracted based on different directions. The local histogram derivative $h_\theta(x, y)$ in the direction θ is defined in Equation (3), where pixels around pixel (x, y) in a local area are divided into two groups based on the boundary with direction θ. Here $l_{\theta,i}$ and $r_{\theta,i}$ are the occurrences for the pixels located in the bin i ($i = 1, 2, \ldots, 32$ based on intensity) from the two groups, respectively.

$$h_\theta(x, y) = \tfrac{1}{2} \sum_i \frac{(l_{\theta,i} - r_{\theta,i})^2}{l_{\theta,i} + r_{\theta,i}} \tag{3}$$

However, the local features from derivatives contain high responses on non-edge points affected by noise or textures. Techniques for manipulating these local features are useful to improve the detection performance. For instance, surround

suppression can reduce some texture responses on image gradients [8]. Therefore, construction of features from basic local features can potentially improve the detection performance.

2.2 Related Work to GP for Edge Detection

GP has been used to choose raw pixels to construct low-level edge detectors. Poli [16] suggested using four macros for searching a pixel's neighbours in image processing using GP, and Ebner [4] approximated the Canny detector by using four shifting functions and other functions in GP. Search operators based on a modified shifting function were used in GP for constructing edge detectors based on ground truth without using windows, and different fitness functions were investigated to evaluate edge detectors [5,6]. A 13×13 window filter was evolved to compare with the Canny edge detector in [19]. A 4×4 window was employed to evolve digital circuits for edge detection by GP [7]. Harris and Buxton [9] designed edge detectors based on the approximation of the responses on one-dimensional signals by GP. All these GP edge detectors are based on raw pixels and their output can be considered as R in Fig. 1.

Also, some image operators have been used to extract features in GP. Morphological operators (erosion and dilation) as terminals were utilised in Linear GP for detecting binary image edges [18]. A rotation variant GP feature constructed based on image filters was used to train a logistic regression classifier with texture gradients in boundary detection [10]. However, only one composite feature (combined with texture gradients) in their work was presented to compete with other edge and contour detectors. The variant feature is based on multiple directions, which leads to high computational cost. In addition, the GP feature needs to be combined with other existing approaches to perform boundary detection.

In summary, there has not been much research for constructing composite features (based on existing features) using GP. Although only one solution in [10] is found in the response computation phase, it makes automatic feature construction in the feature manipulation phase appealing for edge detection.

3 Constructing Variant Features Using GP

3.1 Terminal Set

Variant features (by definition) are dependent on a direction to do extraction, and the number of features is generally large. A variant feature method usually extracts features based on four different directions, namely $\theta = 0°, 45°, 90°, 135°$. For further constructing features by GP, the basic features in the terminal set of the proposed tree-based GP system will only contain variant features based on the fixed direction $\theta = 0°$, so the terminal set size is reduced and the GP system can easily choose these features to construct a new variant feature based on $\theta = 0°$.

In this study, only three variant (horizontal) features are used to construct new features: the image Gaussian derivative $T_{gd,0°}$; a feature $T_{f,0°}$ based on

an F-test on the two groups of pixels separated by a horizontal line; and the histogram derivative $T_{hd,0^\circ}$ [12]. Since the three features are totally different, it is possible to construct new features for improving detection performance. In addition, we also use random constants rnd in the range of $[-100, 100]$ in the terminal set. Here, only image grayscales are used.

3.2 Primitive Functions

The function set contains the four arithmetic functions $\{+, -, *, \div\}$ and three logical operators $\{IF, <, >\}$. Here, $\div$ is protected division, producing a result of 1 for a 0 divisor. IF contains three arguments, and the first one is a boolean. IF will return the second argument with a real number when the first is true, otherwise will return the third argument with a real number. It is possible that a feature is better than another feature for some edge responses, but worse for other responses. The logical operators can be used to only select a good part from one feature, and GP will choose partial responses from one feature.

3.3 Fitness Function

Since we only use the three features based on the horizontal direction, the horizontal true edges from ground truth need to be extracted. Pixels on the other directional (true) edges or on non-edges are considered as negative labels. The output (o_θ) of a program is not the final output for a constructed feature. We employ a simple Bayesian model [3] to map the scale of o_θ into the range of $[0, 1]$ (as the final output $T_{GP,\theta}$). Here the output o_θ is considered as following a Gaussian distribution, which is similar to the work in [17]. Formula (4) presents the weight value p_j (probability) for non-edge points ($j = 0$) or edge points ($j = 1$), where, P_j is the prior probability of each class (using 0.5 for each class in this paper), and $\hat{\mu}_j$ and $\hat{\Sigma}_j$ are the estimated mean(s) and estimated covariance matrix. For the output o_θ, the estimated covariance matrix is only one element, namely the estimated standard deviation from each class. When p_1 is larger than p_0, the output o_θ is considered as an edge point (in a soft edge map), otherwise, a non-edge point. The soft output $T_{GP,\theta}$ for a constructed feature is defined in formula (5).

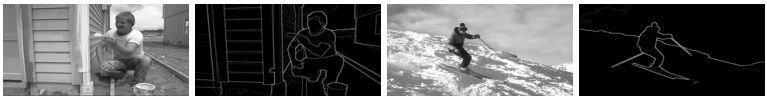
$$p_j = \frac{P_j}{|2\pi\hat{\Sigma}_j|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(o_\theta - \hat{\mu}_j)^T \hat{\Sigma}_j^{-1}(o_\theta - \hat{\mu}_j)\right) \tag{4}$$

$$T_{GP,\theta} = \begin{cases} \frac{p_1}{p_0+p_1} & \text{if} \quad p_1 > p_0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We treat the edge detection task as a binary classification task (with the edge pixels as the main class) in the evolutionary training process. Therefore, the accuracy (the number of pixels correctly discriminated as a proportion of the total number of pixels in the training data) is used as the fitness function. The fitness function is based on the output $T_{GP,\theta}$ without post-processing, following the suggestion from [13].

4 Experiment Design

The Berkeley Segmentation Dataset (BSD) [12] consists of natural images (of size 481×321 pixels) with ground truth provided. All images are independent and are taken throughout the world. The training dataset contains 200 images and the test dataset has 100 images. The ground truth are combined from five to ten persons as graylevel images for fairness of judgement of edges. Fig. 2 shows two example training images and their ground truth. For simplicity, we sample image pixels with the same ratio of edge points and non-edge points as our training data. We mark the horizontal edge points based on three straight connected edge points in the horizontal direction as positive labels, and the others are negative labels. Approximately 125 horizontal edge points, 125 non-horizontal edge points and 250 non-edge points are randomly sampled from each training image. Therefore, the training data includes approximately $100,000$ $((125 + 125 + 250) * 200$ images) cases and the three features in the horizontal direction. The window size for $T_{gd,0^\circ}$, $T_{f,0^\circ}$ and $T_{hd,0^\circ}$ is 7×7.



(a) 23080.jpg (b) 61060.jpg

Fig. 2. Two example images from BSD Training dataset and their ground truth

The parameter values for GP are: population size 500; maximum generations 200; maximum depth (of a program) 8; and probabilities for mutation 0.15, crossover 0.80 and elitism (reproduction) 0.05. These values are chosen based on common settings and initial experiments. There are 30 independent runs.

In order to test new variant features constructed by GP, we apply these new horizontal features to the vertical direction, and then combine the outputs from the horizontal and vertical directions together by the square root of sum of squares as a final detection result. Note that when we apply a new horizontal feature $T_{GP,0^\circ}$ to the vertical direction $T_{GP,90^\circ}$, the basic features ($T_{dg,0^\circ}$, $T_{f,0^\circ}$, $T_{hd,0^\circ}$) are changed to the relevant outputs ($T_{dg,90^\circ}$, $T_{f,90^\circ}$, $T_{hd,90^\circ}$) from 90°.

To measure the performance of these features constructed by GP, the F-measure is used in the *test* phase [2,12]. The F-measure (used in [2,12] as $F = \frac{2\text{recall}*\text{precision}}{\text{recall}+\text{precision}}$) is the combination of recall (the number of pixels on the edges correctly detected as a proportion of the total number of pixels on the edges) and precision (the number of pixels on the edges correctly detected as a proportion of the total number of pixels detected as edges). In the F-measure evaluation system, pixels are discriminated as edge points based on the value of their features larger than a threshold, and the predicted edges are simply thinned by the thinning operator [12]. After obtaining thinned prediction, an optimal matching operator will be used to match the prediction and the ground truth. Based on different threshold level indices $k = 0, 1, ..., 51$, a maximum F_{max}

Table 1. Comparison of F_{max} values among constructed features by GP, Image Gaussian Derivatives T_{gd}, F-tests T_f, Histogram Derivatives T_{hd}, and a Bayesian Model for the 100 BSD test images

	F_{max}
GP	0.5776 ± 0.0015
T_{gd}	0.4737
T_f	0.5489
T_{hd}	0.542
Bayesian	0.5274

($F_{max} = \max\{F_k\}$) will be considered as the measurement for the feature, where F_k is the F value when the threshold level index k is used. The threshold value at the threshold index level k is $\frac{k}{52}$.

For fair comparison, the F_{max} values of the three basic features (after normalisation) also are given without post-processing, so their values are different from the final performance evaluation in [12]. The final detection result from a variant feature are selected from the maximum outputs from $\theta = 0°, 45°, 90°, 135°$, and we use T_{gd}, T_f and T_{hd} to indicate the final results from Gaussian derivatives, F-tests and histogram derivatives, respectively.

5 Results and Discussion

5.1 Overall Results

Table 1 presents the mean and standard deviation of F_{max} values of the results from the combination of 30 features constructed by GP and their rotated versions $T_{GP,90°}$, and F_{max} values from T_{gd}, T_f, T_{hd}. Also, an estimated Bayesian model (using formulae (4) and (5)) [3] based on the sampling dataset is used to extract features in 0° for the test images when $T_{gd,0°}$, $T_{f,0°}$, $T_{hd,0°}$ are considered as independent variables, replacing o_θ. The detection results for the estimated Bayesian model are the square root of sum of squares of the horizontal and vertical directions. The training time for each constructed feature is around 11 hours but the testing time is very short (several milliseconds). Here, C++ is used for the implementation on a single machine with CPU 3.1 GHz. Note that the three basic features are pre-calculated and ready to use for testing.

The results from GP are significantly better than the others based on the one sample t-test with significance level 0.05. The test results show that the variant features constructed by GP significantly improve the detection performance based on the combination in the horizontal and vertical directions. However, the combination of the three features by the Bayesian model does not improve the detection performance. Only using the horizontal response information to estimate a Bayesian model, the estimated model has worse detection results than the results from T_f and T_{hd}. Therefore, we can see that GP is effective for automatic construction of variant features, only using one directional edge information.

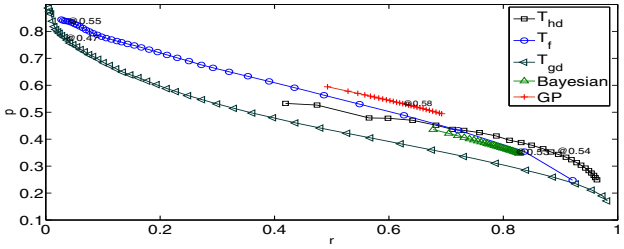


Fig. 3. Details for recall and precision of T_{gd}, T_f, T_{hd} and GP (average)

From Table 1, we can see that the standard deviation of the detection results from GP is very low. Therefore, the evolved variant features have good stability to do edge feature extraction, although they are trained based only on the horizontal edge information.

5.2 Comparison among GP, T_{gd}, T_f and T_{hd}

Fig. 3 shows the details for recall and precision with different threshold levels. Here, "@" is the position for the F_{max}. Compared with the detection from T_{gd}, T_f and T_{hd}, the curve for the averages of recall and precision of the detection results from GP is obviously better than the three basic curves from T_{gd}, T_f and T_{hd}. Therefore, the recall vs precision curves also show that GP can construct good variant features based on fixed directional edge information. Based on formula (5) ($T_{GP,\theta}$), the 30 features constructed by GP do not consider responses with low probabilities for discriminating pixels as edge points (p_1 is not larger than p_0), which is a reason that the curve for GP is short (by only showing the points with both precision and recall higher than 0.5).

5.3 Detected Images

Fig. 4 visually presents some detection results from GP, T_{gd}, T_f, T_{hd} and the estimated Bayesian model, where "GT" is ground truth. The five example images presented from the BSD test images contain very different content. Firstly, it is found that the detected results from T_{gd} are affected by noise and textures, such as the responses on the wall texture in image 385039. Secondly, most of the detected results from T_f are correct, but some have weak responses on edges in low contrast areas, such as the boundaries of the objects in images 62090 and 106024. Thirdly, histogram derivatives T_{hd} have stronger responses on edges in low contrast areas, but overreact to tiny discontinuities (because of overweighting discontinuities from local histograms), such as the water wave in image 101087.

The variant feature constructed by the Bayesian model improves responses on edges in low contrast areas, such as the boundary of the object in image 69020, but still has problems existing in the basic features, such as the responses on the walls in image 385039. However, the detected results from GP present very good detection performance on these images. The constructed feature enhances

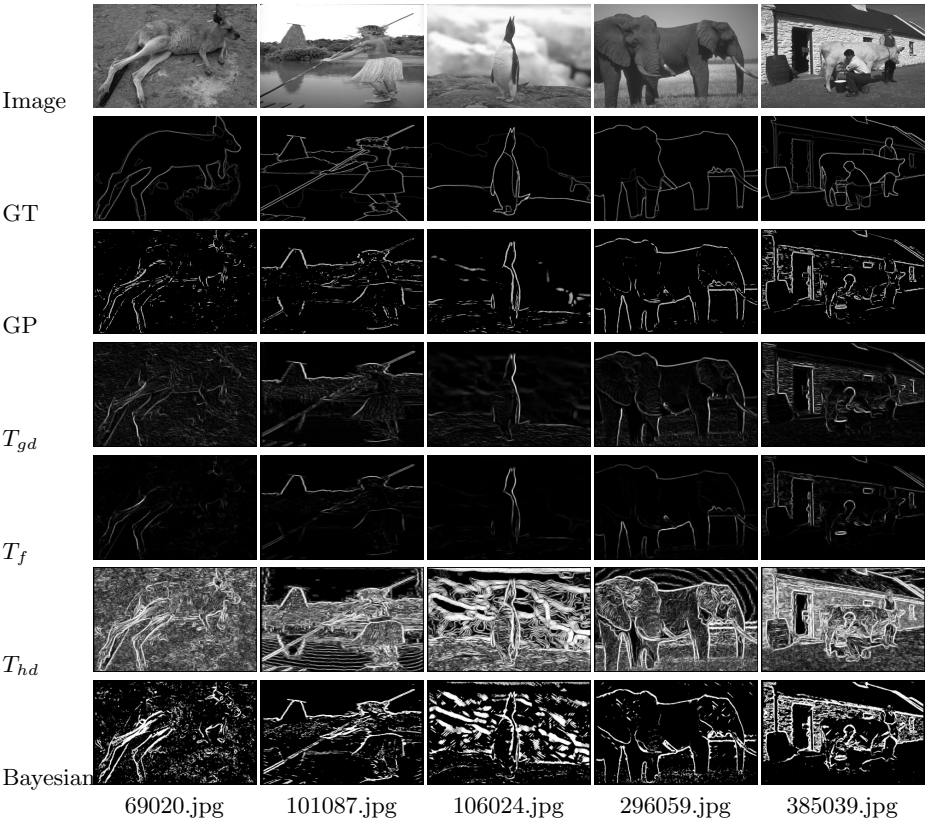


Fig. 4. Detected images based on the different features

responses on edges in low contrast areas, such as the boundary of the object in image 69020. Also it suppresses noise and textures, such as the background in image 69020. From these detected images, the feature constructed by GP has the advantages from the basic features, and avoids some disadvantages from them. Therefore, these examples confirm the effectiveness of the GP method.

6 Conclusions

The goal of this paper was to investigate using GP to construct variant features for edge detection to improve the detection performance. The goal was successfully achieved based on construction of variant features in a fixed direction. Three variant features, namely image Gaussian derivatives, F-test and image local histogram derivatives were used to construct variant GP features. Computational results suggest these features are significantly better than the basic features alone on the BSD test image dataset (based on the F-measure). Also, the comparison between GP and a simple Bayesian model shows that GP has ability to find a way of effectively combining different features together.

For future work, we will test this technique on a larger set of basic features and analyse the complexity of evolved programs. In addition, other machine learning algorithms will be used to compare with GP. Post-processing techniques will be employed to obtain the final solutions, and the final edge maps will be compared to state-of-the-art edge and contour detectors.

References

1. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)
2. Dollar, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1964–1971 (2006)
3. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience (2000)
4. Ebner, M.: On the edge detectors for robot vision using genetic programming. In: Proceedings of Horst-Michael Groβ, Workshop SOAVE 1997 - Selbstorganisation von Adaptivem Verhalten, pp. 127–134 (1997)
5. Fu, W., Johnston, M., Zhang, M.: Genetic programming for edge detection using blocks to extract features. In: Genetic and Evolutionary Computation Conference, pp. 855–862 (2012)
6. Fu, W., Johnston, M., Zhang, M.: Genetic programming for edge detection via balancing individual training images. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2012)
7. Golonek, T., Grzechca, D., Rutkowski, J.: Application of genetic programming to edge detector design. In: Proceedings of the International Symposium on Circuits and Systems, pp. 4683–4686 (2006)
8. Grigorescu, C., Petkov, N., Westenberg, M.A.: Contour and boundary detection improved by surround suppression of texture edges. Image and Vision Computing 22(8), 609–622 (2004)
9. Harris, C., Buxton, B.: Evolving edge detectors with genetic programming. In: Proceedings of the First Annual Conference on Genetic Programming, pp. 309–314 (1996)
10. Kadar, I., Ben-Shahar, O., Sipper, M.: Evolution of a local boundary detector for natural images via genetic programming and texture cues. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1887–1888 (2009)
11. Lim, D.H., Jang, S.J.: Comparison of two-sample tests for edge detection in noisy images. Journal of the Royal Statistical Society. Series D (The Statistician) 51(1), 21–30 (2002)
12. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 530–549 (2004)
13. Moreno, R., Puig, D., Julia, C., Garcia, M.: A new methodology for evaluation of edge detectors. In: Proceedings of the 16th IEEE International Conference on Image Processing (ICIP), pp. 2157–2160 (2009)
14. Papari, G., Campisi, P., Petkov, N., Neri, A.: A biologically motivated multiresolution approach to contour detection. EURASIP Journal on Applied Signal Processing 2007, 119–119 (2007)

15. Papari, G., Petkov, N.: Edge and line oriented contour detection: state of the art. Image and Vision Computing 29, 79–103 (2011)
16. Poli, R.: Genetic programming for image analysis. In: Proceedings of the First Annual Conference on Genetic Programming, pp. 363–368 (1996)
17. Smart, W., Zhang, M.: Probability Based Genetic Programming for Multiclass Object Classification. In: Zhang, C., Guesgen, H.W., Yeap, W.-K. (eds.) PRICAI 2004. LNCS (LNAI), vol. 3157, pp. 251–261. Springer, Heidelberg (2004)
18. Wang, J., Tan, Y.: A novel genetic programming based morphological image analysis algorithm. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 979–980 (2010)
19. Zhang, Y., Rockett, P.I.: Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp. 795–802 (2005)

Automatic Construction of Gaussian-Based Edge Detectors Using Genetic Programming

Wenlong Fu[1], Mark Johnston[1], and Mengjie Zhang[2]

[1] School of Mathematics, Statistics and Operations Research Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
[2] School of Engineering and Computer Science
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
wenlong.fu@msor.vuw.ac.nz, {mark.johnston,mengjie.zhang}@vuw.ac.nz

Abstract. Gaussian-based edge detectors have been developed for many years, but there are still problems with how to set scales for Gaussian filters and how to combine Gaussian filters. In order to address both problems, a Genetic Programming (GP) system is proposed to automatically choose scales for Gaussian filters and automatically combine Gaussian filters. In this study, the GP system is utilised to construct rotation invariant Gaussian-based edge detectors based on a benchmark image dataset. The experimental results show that the GP evolved Gaussian-based edge detectors are better than the Gaussian gradient and rotation invariant surround suppression to extract edge features.

Keywords: Genetic Programming, Edge Detection, Gaussian Filter.

1 Introduction

Edge detection is a well developed area of image analysis [1,20]. Generally, the goal of edge detection is to find discontinuities between different regions or between background and objects. Many methods have been proposed for detecting edges in images [1,10,20]. However, edge detection is a subjective task and the suitability of a solution for one image is normally dependent on human observation, which makes automatically constructing edge detectors for special tasks more appealing.

Gaussian-based edge detection techniques have been developed for many years, and some advantages for detecting edges exist in these techniques [1]. Different Gaussian-based techniques have been developed, such as Difference of Gaussians (DoG) and Laplacian of Gaussian (LoG) [18], the Canny edge detector [5], and surround suppression [13]. Multiple Gaussian filters can be combined to improve detection performance [1,20]. However, how to choose scales for Gaussian filters and how to combine multiple Gaussian filters still need to be investigated [1].

Genetic Programming (GP) has been employed for edge detection since at least 1996 [14,21]. GP has been used to evolve low-level edge detectors [8,9]. Since GP can automatically construct edge detectors, it would be interesting to utilise GP to evolve Gaussian-based edge detectors, i.e., where a GP system

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 365–375, 2013.
© Springer-Verlag Berlin Heidelberg 2013

selects the scale of the Gaussian filters and combines different Gaussian filters to perform edge detection.

The goal of this paper is to investigate automatic construction of rotation invariant Gaussian-based edge detectors using GP. In order to address the scale setting problem for a Gaussian filter, the proposed GP system needs to automatically choose a good scale for a Gaussian filter used to construct edge detectors. To address the combination problem for Gaussian filters, the proposed GP system needs to construct new edge detectors based on the selected Gaussian filters with different scale values. Specifically, we investigate the following research objectives: (1) how to design a GP system so that GP can automatically choose Gaussian filters with different scales and combine them as rotation invariant edge detectors; (2) whether evolved Gaussian-based edge detectors can compete with some existing Gaussian-based edge detectors designed by humans; and (3) what characteristics of GP edge detectors can be obtained from visual results detected by these edge detectors.

In the remainder of the paper, Section 2 briefly describes some relevant background. Section 3 presents how GP can be used to construct invariant Gaussian-based edge detectors. After presenting the experimental design in Section 4, Section 5 describes the results with discussions. Section 6 draws conclusions and suggests future work directions.

2 Background

2.1 Gaussian-Based Edge Detection

Gaussian filters are widely used as smoothing filters, playing an important role in edge detection in human visual systems [1]. Gaussian-based filters have been developed based on a combination of smoothing and differentiation. The different responses on edges from different scales of Gaussian filters are also useful for detecting boundaries between two different regions, or between objects and background [18,23].

In order to filter noise, methods based on approximation to the shape of spatial receptive fields employ Gaussian filters and differentiation to perform edge detection, such as DoG [1]. Given a Gaussian filter $g_\sigma(x, y)$ (see Equation (1), where σ is a *scale* parameter), the DoG is defined in Equation (2), where σ_1 and σ_2 are different scale parameters. DoG is a second derivative filter [24], approximating LoG well. DoG, as a kind of band filter, suppresses noise with a high spatial frequency, but decreases overall image contrast [18].

$$g_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{1}$$

$$DoG_{\sigma_1,\sigma_2}(x, y) = g_{\sigma_1}(x, y) - g_{\sigma_2}(x, y) \tag{2}$$

Canny detectors [5] are derived from an optimal filter based on the local maxima resulting from the convolution of a filter with the signal affected by white noise in one dimension, which is approximated by the derivative of a Gaussian function

[1]. Canny edge detectors use adaptive thresholding with hysteresis to eliminate breaking of edge contours, but they are slightly sensitive to weak edges and susceptible to spurious and unstable boundaries with non-significant change in intensity [1,20].

Gaussian filters are often combined to detect edges based on responses at different scales. The basic premise of using multi-Gaussian filters (multi-scale edge detectors) is that different areas of an image have varying noise and edge types; therefore a special filter is used to smooth a relevant area of the image. There are three directions to use the multi-scale technique. The first proceeds from a coarse solution to a fine solution, namely edge focusing [3]. In this method, a large scale (high σ) Gaussian filter is used to detect edges, and then the next smaller scale is used to find the locations of edges. Multi-scale Gaussian filters are used to reverse the effect of the blurring caused by large scale Gaussian filters. However, how to set each level scale and choose the threshold in each level is hard. The threshold at the coarsest level determines the detected edge quality. The second is from fine to coarse [17]. When coarse solutions are used to detect edges, the localisation error problem still exists. Again, how to choose scales is not clear. The third direction is to use adaptive Gaussian filters (whose scales are adapted to both the noise characteristics and the local variance of a small area in an image) to detect edges [2]. Assuming that noise can be modelled by a Gaussian distribution with a known variance, this method smooths areas using a large scale to filter out noise. For real images, the noise variance has to be estimated.

Over the most recent ten years, contextual information has been used in Gaussian-based edge detection. In order to detect edges and filter noise, surround suppression [12,13] has been developed. In this technique, an operation, called *inhibition*, is used to suppress responses on textures via combining the responses from a DoG and the gradient of a Gaussian filter. In surround suppression, Gabor filters are normally used [13]. Since a two-dimensional Gabor filter is the product of a Gaussian kernel function and a sinusoidal function, and Gaussian filters can replace Gabor filters in this method, the surround suppressing technique can still be considered as a kind of Gaussian-based edge detection. The main benefit of surround suppression is to filter noise caused by textures [20].

2.2 Related Work to GP for Edge Detection

There has not been much previous work done using GP for edge detection. Harris and Buxton [14] designed approximate response detectors in one-dimensional signals with GP. Poli [21] suggested using four macros for searching a pixel's neighbours in image processing using GP, and Ebner [7] used four shifting functions and other functions to approximate the Canny detector by GP. The Sobel detector was approximated by hardware design [15] with the relationship between a pixel and its neighbourhood as terminals. Bolis et al [4] used GP to evolve programs to search edges in images. Zhang and Rockett [26] evolved a 13×13 window filter for comparison with the Canny edge detector. A 4×4 window was used to evolve digital circuits (combination of bit operators or gates)

for edge detection by GP [11]. Ground truth and a shifting function were used to evolve low-level edge detectors by GP without using windows [8,9].

Also, some image operators have been used to extract features in GP. Wang and Tan [25] used Linear GP to find edges, inspired by morphological operators (erosion and dilation) as terminals [22], for binary images. GP was used to construct rotation variant features with image filters in different directions, and these features were combined with texture gradients to train a logistic regression classifier for boundary detection [16]. However, only one solution (combined with texture gradients) in their work was presented to compete with other edge and contour detectors.

In summary, the existing work does not include much research for *constructing* Gaussian-based edge detectors. Since Gaussian-based edge detectors are considered a popular method to detect images [20], it is worth investigating how to employ GP to construct Gaussian-based edge detectors.

3 Constructing Invariant Gaussian-Based Edge Detectors Using GP

3.1 Terminals Based on Gaussian Models

To rapidly find a Gaussian-based edge detector, the terminal set in our proposed GP system includes the gradient of a Gaussian filter, LoG, and DoG. The gradient magnitude of a Gaussian filter is shown in Equation (3). LoG is given in Equation (4). DoG is shown in Equation (2).

$$\frac{\partial g(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$\frac{\partial g(x,y)}{\partial y} = -\frac{y}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$\nabla g(x,y) = \sqrt{\left(\frac{\partial g(x,y)}{\partial x}\right)^2 + \left(\frac{\partial g(x,y)}{\partial y}\right)^2} \tag{3}$$

$$\nabla^2 g(x,y) = \frac{x^2+y^2-2\sigma^2}{2\pi\sigma^6}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{4}$$

The terminal set also includes random constants *rnd* (real numbers) in the range of $[-10, 10]$. In this terminal set, the parameter σ is randomly generated in the range of [1..5]. Let the large scale in the DoG be *double* the small one, the scale range of all Gaussian filters is from 1 to 10. Therefore, the coarsest scale (from 2 to 10) covers the range of [3..6] as suggested in [3] so that it is possible to find more Gaussian filters.

3.2 Function Set

Since Gaussian filters in the terminal set can be considered as edge detectors, we choose a simple function set, namely $\{+, -, \times, \div, \circledast\}$. Here, $\div$ is protected

division, producing a result of 1 for a 0 divisor; and $\circledast$ is a modified convolution based on the surround suppression operation, which takes two arguments. The second argument $f(x, y)$ of $\circledast$ will be inhibited as in the formulae (5) and (6), where $f(x, y)$ is an output from a Gaussian filter or a temporary output from a subtree. The first argument of $\circledast$ will be convolved with $s(h(f(x, y)))$, producing an output with the first argument's size, e.g., the first argument could be $\nabla g(x, y)$. The transformation $h(f(x, y))$ removes the negative values from neighbours, and the transformation $s(h(f(x, y)))$ will perform normalisation so that the output of $\circledast$ has the same range as the first argument of $\circledast$. The output of $\circledast$ is an inhibited term. For further details about this operation, see [13]. An existing surround suppression technique can be expressed by the GP system as $\nabla g(x, y) - C_1 * \nabla g(x, y) \circledast DoG_{\sigma_1,\sigma_2}(x, y) - C_2$, where C_1 and C_2 are constants.

$$h(f(x, y)) = \max\{f(x, y), 0\} \tag{5}$$

$$s(h(f(x, y))) = \frac{h(f(x, y))}{\parallel h(f(x, y)) \parallel} \tag{6}$$

3.3 Fitness Function

Figure of Merit (FOM) has been investigated as a fitness function to evaluate edge detection performance based on both localisation and accuracy [9]. FOM can be used as a fitness function in GP to evolve low-level edge detectors. We choose FOM based on each training image (not the overall pixels) as the fitness function in the GP system. FOM is defined in Equation (7), where M is the number of training images, $N_{i,T}$ is the number of true edge points in image i, $N_{i,P}$ is the number of predicted edge points in image i, $Set_{i,P}$ is the set of all predicted edge points for image i, α is a weighting factor for detection localisation, and $d(j)$ is the distance from a predicted edge point j to the nearest true edge point in a ground truth edge map. Considering the overlap of a 3×3 window, α is usually set $\frac{1}{9}$.

$$FOM = \frac{1}{M} \sum_{i=1}^{M} \left(\frac{1}{\max\{N_{i,T}, N_{i,P}\}} \sum_{j \in Set_{i,P}} \frac{1}{1 + \alpha d^2(j)} \right) \tag{7}$$

4 Experiment Design

The Berkeley Segmentation Dataset (BSD) [19] consists of natural images (of size 481×321 pixels) with ground truth provided. All images are independent and are taken from throughout the world. The training dataset contains 200 images and the test dataset has 100 images. For fairness of judgement of edges, the ground truth are combined from five to ten persons as graylevel images. Since there are redundancies existing in the training data and the computational cost is high for GP, we only select 20 images with rich edge contents as the training data. Since a candidate including Gaussian filters is expected to have some ability to

23080.jpg 41004.jpg 61060.jpg 106020.jpg 207056.jpg 216053.jpg

Fig. 1. Six example training images from BSD dataset and their ground truth

detect images, it is possible to find which candidates from a set of candidates generated by the GP system are good at edge detection based on the small set of images. Figure 1 shows six example training images (from the 20 images) and their ground truth. The pixels with graylevel 0 (dark) in the ground truth are non-edge points, and the others are edge points.

The parameter values for GP are: population size 200; maximum generations 200; maximum depth (of a program) 7; and probabilities for mutation 0.15, crossover 0.80 and elitism (reproduction) 0.05. These values are chosen based on common settings and initial experiments. We perform 30 independent runs for the experiment.

The test performance evaluation is directly based on the binary outputs of GP edge detectors using the fixed threshold 0 (positive values for edge points, otherwise for non-edge points), without non-maximum suppression post-processing. To measure the performance of GP edge detectors, the popular F-measure is used in the testing phase [6,19] based on an optimal matching between predicted edge points and ground truth after thinning predicted edges. The F-measure (used in [19,6] as $F = \frac{2\text{recall}*\text{precision}}{\text{recall}+\text{precision}}$) is the combination of recall (the number of pixels on the edges correctly detected as a proportion of the total number of pixels on the edges) and precision (the number of pixels on the edges correctly detected as a proportion of the total number of pixels detected as edges).

5 Results and Discussion

5.1 Overall Results

Table 1 gives the mean and standard deviation of the F values on the 100 BSD test images, and the averages of recall and precision from 30 evolved Gaussian-based edge detectors. Also, the detected results from the Sobel edge detector, the Gaussian gradient (GG), and (rotation invariant) surround suppression (SS) are given in the table for comparison purposes. Since how many filters from each terminal (with different parameters) should be chosen to combine as a composite filter is not clear, the combination technique only selects SS to compare with the evolved results. We also perform a statistical comparison among these edge

Table 1. F values and means of recall and precision of GP Gaussian-based edge detectors, the Sobel detector, the Gaussian Gradient (GG), and Surround Suppression (SS) on the BSD test image dataset (100 images)

Training	F	recall	precision
GP	0.5628 ± 0.0131	0.6681	0.4893
Sobel	0.4833	0.6035	0.4030
GG	0.5153	0.6221	0.4399
SS	0.5381	0.6747	0.4475

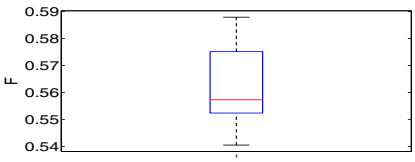


Fig. 2. Boxplot of 30 GP edge detectors' performances on the BSD test image dataset

detector performances based on F values. The evolved edge detectors are significantly better than the Sobel edge detector, the Gaussian gradient, and the invariant surround suppression method to detect edges on the BSD test images (based on one sample t-tests with significance level 0.05). Also, the average precision from the GP edge detectors is the highest among these edge detectors, and the average recall is only slightly lower than that from surround suppression. Therefore, the GP Gaussian-based edge detectors have good performances on recall and precision.

From the boxplot in Figure 2, we can see that the F value of the best Gaussian-based edge detector from these evolved edge detectors is close to 0.59, and the F value of the worst is higher than 0.54. All 30 edge detectors evolved by GP have a higher F than the other three edge detectors in Table 1. Also, more than 75% of these evolved edge detectors have F values higher than 0.55. Therefore, the evolved edge detectors have reliable detection performance on the BSD test images even though we only select 20 images from the original 200 training images.

In summary, the evolved edge detectors have high performance (the mean of F values) and reliability (the worst evolved edge detector has a higher F than the other three edge detectors in Table 1). A potential reason for this good performance is that any generated candidate including different scales of Gaussian filters has some ability to detect edges. The training data (20 images) can be used to find the candidates with good scales that GP can automatically choose and good structures that GP can automatically combine.

5.2 Detected Images

To visually compare the detected images, Figure 3 shows five example images from the BSD test image dataset, and the relevant detected (binary) results

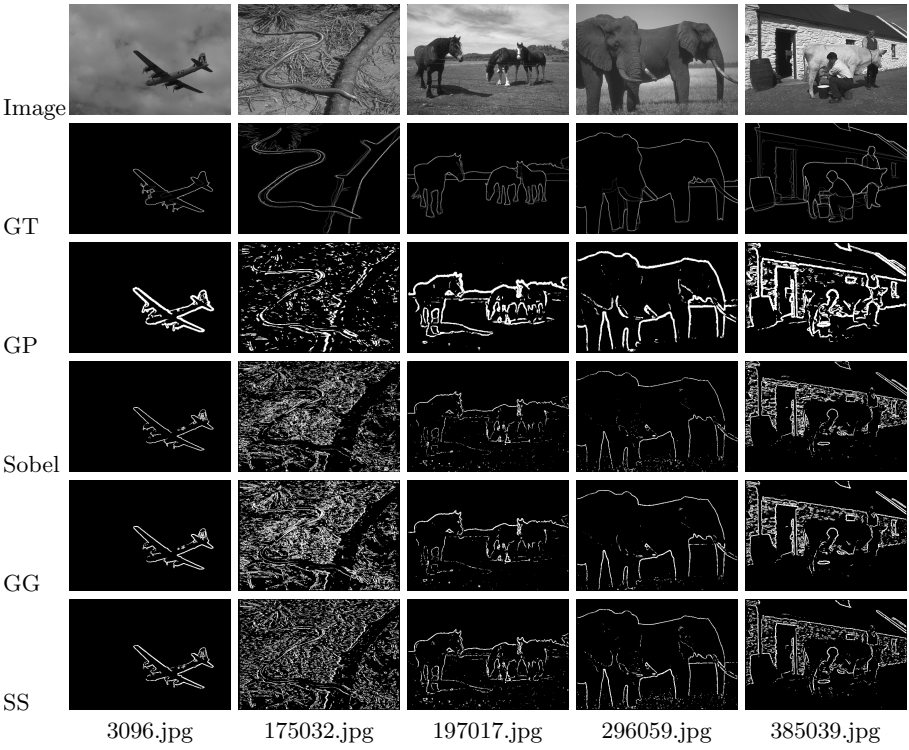


Fig. 3. Detected images by the best GP edge detector, the Sobel edge detector, the Gaussian gradient (GG) and surround suppression (SS). Note that GT is ground truth.

by the best GP edge detector, the Sobel edge detector, GG and SS. Note that the soft edge maps from the existing edge detectors are normalised in the range between 0 to 1. The binary edge maps from the soft edge maps are obtained after using the best thresholds (from $\frac{k}{52}, k = 0, 1, 2, ..., 51$, based on the maximum F on the 100 images). The best thresholds for the Sobel edge detector, GG and SS are 0.2651, 0.2793 and 0.0392, respectively.

Considering images 3096 and 296059, the Sobel edge detector and Gaussian gradient miss some true edges in both images, such as the middle of the plane (top side) in image 3096, and two lines in the middle right of image 296059. Note that both edge detectors can detect the middle of the plane, but the relevant threshold is lower than the best threshold on the 100 BSD test images. Surround suppression detects almost all of the middle part of the plane, and the GP edge detector fully detects the middle part of the plane. In image 296059, the GP edge detector finds most of true edge points, and it is very weakly affected by noise, but the other three edge detectors miss some information from the background, and they are affected by noise. Also, the GP edge detector presents good detection performance on image 197017. Therefore, the GP evolved edge detector has good detection performance on the BSD test images with few textures.

Comparing the detection results from images with complex textures, such as image 175032 including irregular sticks, and image 385039 including a strong wall texture, the GP edge detector suppresses most of these textures (few responses on irregular sticks and wall textures), but the other detectors are affected by these textures. Therefore, the evolved edge detector shows good performance to suppress strong responses on textures.

However, the detected edges from GP are thicker than the others based on the detected images in Figure 3. A potential reason is that large scale Gaussian filters exist in the GP edge detectors. Non-maximum suppression could be used to thin the detected results. Another reason is that the fitness function FOM allows overlap in detected edges. Also, the training time for each run (for 20 images) is around four and half days on a single machine with CPU 3.1GHz (using C++), so this is a time-consuming process, but this can be sped up by using a computational grid. All GP evolved edge detectors take less than half a second to detect a BSD image. Simplification on these evolved detectors can make this detection time even shorter, which will be investigated in the future.

6 Conclusions

The goal of this paper was to investigate using GP to automatically construct rotation invariant Gaussian-based edge detectors. Based on a proposed GP system using Gaussian filters (with variant scale) as terminals and a modified convolution as a function, the goal was successfully achieved by evolving Gaussian-based edge detectors on a benchmark image dataset. The evolved Gaussian-based edge detectors significantly outperformed the Sobel edge detector, the Gaussian gradient, and invariant surround suppression. The visual results also show that the evolved edge detector has good ability to reject noise and suppress textures.

For future work, a dynamic threshold technique will be introduced in the GP system so that edge detectors with good combinations (but not good detection when using threshold 0) will be found. Also, the system has very high computational training cost, so we will investigate ways to reduce computational cost for the system, while maintaining good performance. In additional, we will analyse the complexity for the evolved edge detectors.

References

1. Basu, M.: Gaussian-based edge-detection methods: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 32(3), 252–260 (2002)
2. Bennamoun, M., Boashash, B., Koo, J.: Optimal parameters for edge detection. In: Proc. of IEEE Int. Conference on Systems, Man and Cybernetics, vol. 2, pp. 1482–1488 (1995)
3. Bergholm, F.: Edge focusing. IEEE Transactions on Image Processing 9, 726–741 (1987)

4. Bolis, E., Zerbi, C., Collet, P., Louchet, J., Lutton, E.: A GP Artificial Ant for Image Processing: Preliminary Experiments with EASEA. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tetamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 246–255. Springer, Heidelberg (2001)
5. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)
6. Dollar, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1964–1971 (2006)
7. Ebner, M.: On the edge detectors for robot vision using genetic programming. In: Proc. of Horst-Michael Groβ, Workshop SOAVE 1997 - Selbstorganisation von Adaptivem Verhalten, pp. 127–134 (1997)
8. Fu, W., Johnston, M., Zhang, M.: Genetic programming for edge detection: a global approach (2011)
9. Fu, W., Johnston, M., Zhang, M.: Genetic programming for edge detection based on figure of merit. In: Proc. of Genetic and Evolutionary Computation Conference, pp. 1483–1484 (2012)
10. Ganesan, L., Bhattacharyya, P.: Edge detection in untextured and textured images: a common computational framework. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 27(5), 823–834 (1997)
11. Golonek, T., Grzechca, D., Rutkowski, J.: Application of genetic programming to edge detector design. In: Proc. of the Int. Symposium on Circuits and Systems, pp. 4683–4686 (2006)
12. Grigorescu, C., Petkov, N., Westenberg, M.: Contour detection based on non-classical receptive field inhibition. IEEE Transactions on Image Processing 12(7), 729–739 (2003)
13. Grigorescu, C., Petkov, N., Westenberg, M.A.: Contour and boundary detection improved by surround suppression of texture edges. Image and Vision Computing 22(8), 609–622 (2004)
14. Harris, C., Buxton, B.: Evolving edge detectors with genetic programming. In: Proc. of the First Annual Conference on Genetic Programming, pp. 309–314 (1996)
15. Hollingworth, G., Smith, S., Tyrrell, A.: Design of highly parallel edge detection nodes using evolutionary techniques. In: Proc. of the Seventh Euromicro Workshop on Parallel and Distributed Processing, pp. 35–42 (1999)
16. Kadar, I., Ben-Shahar, O., Sipper, M.: Evolution of a local boundary detector for natural images via genetic programming and texture cues. In: Proc. of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1887–1888 (2009)
17. Lacroix, V.: The primary raster: a multiresolution image description. In: Proc. of the 10th Int. Conference on Pattern Recognition, vol. I, pp. 903–907 (1990)
18. Marr, D., Hildreth, E.: Theory of edge detection. Proc. of the Royal Society of London, Series B, Biological Sciences. 207, 187–217 (1980)
19. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 530–549 (2004)
20. Papari, G., Petkov, N.: Edge and line oriented contour detection: state of the art. Image and Vision Computing 29, 79–103 (2011)
21. Poli, R.: Genetic programming for image analysis. In: Proc. of the First Annual Conference on Genetic Programming, pp. 363–368 (1996)
22. Quintana, M.I., Poli, R., Claridge, E.: Morphological algorithm design for binary images using genetic programming. Genetic Programming and Evolvable Machines 7, 81–102 (2006)

23. Schunck, B.: Edge detection with Gaussian filters at multiple scales. In: IEEE Workshop on Computer Vision, Representation and Control, pp. 208–210 (1987)
24. Song, D.M., Li, B.: Derivative computation by multiscale filters. Image and Vision Computing 16(1), 43–53 (1998)
25. Wang, J., Tan, Y.: A novel genetic programming based morphological image analysis algorithm. In: Proc. of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 979–980 (2010)
26. Zhang, Y., Rockett, P.I.: Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In: Proc. of the Conference on Genetic and Evolutionary Computation, pp. 795–802 (2005)

Implicit Fitness Sharing for Evolutionary Synthesis of License Plate Detectors

Krzysztof Krawiec and Mateusz Nawrocki

Institute of Computing Science, Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland

Abstract. A genetic programming algorithm for synthesis of object detection systems is proposed and applied to the task of license plate recognition in uncontrolled lighting conditions. The method evolves solutions represented as data flows of high-level parametric image operators. In an extended variant, the algorithm employs implicit fitness sharing, which allows identifying the particularly difficult training examples and focusing the training process on them. The experiment, involving heterogeneous video sequences acquired in diverse conditions, demonstrates that implicit fitness sharing substantially improves the predictive performance of evolved detection systems, providing maximum recognition accuracy achievable for the considered setup and training data.

Keywords: Genetic programming, pattern recognition, image analysis, implicit fitness sharing, license plate recognition.

1 Introduction

Manual design of image analysis systems is a time-consuming task that requires a lot of expertise. Even for a skilled expert, the final outcome of a chain of image processing algorithms is hard to predict, so usually many designs have to be laboriously tested to come up with a well performing image analysis system.

In this study we automate this search process using genetic programming [7], allowing the search algorithm to compose complete image analysis programs. The programs maintained in the population are composed of instructions that implement image processing algorithms known from literature. The instructions are allowed to have parameters, which also undergo evolutionary tuning, so overall the method performs search in a joint space of *structures* (data flows) and *parameters* of image analysis programs.

The major contribution of this paper are the experimental outcomes that assess the method with respect to its ability to model the dependencies observable in data (training set performance) and capability of generalization (testing set performance). In particular, we propose to extend the basic approach with implicit fitness sharing, which entices the individuals in population to pay more attention to the particularly difficult examples.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 376–386, 2013.
© Springer-Verlag Berlin Heidelberg 2013

2 Related Work

License plate recognition is one of the best established applications and a common benchmark for pattern recognition and computer vision systems. Former research on this topic engaged various paradigms of computational intelligence, including artificial neural networks, fuzzy logic, and evolutionary computation (see review in [10]). For instance, in [18], a fuzzy logic approach has been applied to this task. In [17], the authors use immune and genetic algorithms to acquire the parameters for the initial step of plate recognition. [15] uses genetic algorithm to optimize weights of neural network that performs the recognition task. In [6], genetic algorithm is used to determine the location of license plate in an image. Other examples of plate recognition systems involving techniques characteristic to computational intelligence can be found in, among others, [1,5,16].

Many of license plate recognition systems successfully implemented in real-world environments assume that vehicles are close to the camera, do not move (or are close to still), and the lighting is at least partially controlled (e.g., infrared emitters or flash light is involved). This is characteristic for deployments like authorization of entry for parking lots and gated blocks-of-flats. However, plate recognition task becomes much more challenging when performed in uncontrolled conditions, which is the case in this study, where the operating conditions resemble more CCTV (closed-circuit television) monitoring in urban areas. Most importantly, the camera used in the experimental part of this paper observes the moving vehicles from a relatively long distance. As a consequence, the observed projected dimensions of the plates are much smaller, and the images can be distorted by motion blur and perspective projection. Also, nothing is assumed about the lighting conditions. We allow also for the presence of multiple vehicles in the field of view.

3 The Approach

We divide the entire task of license plate recognition into four separate stages: motion segmentation, plate detection, character segmentation, and character recognition. Except for motion segmentation, each video frame is processed independently. The stage that undergoes evolutionary learning described in following is plate detection; the remaining stages have been designed manually and remain fixed during evolution. In particular, character recognition is carried out using a support vector machine (SVM, [3,13]), previously trained on a large collection of human-segmented characters belonging to 36 classes (26 uppercase Latin alphabet letters plus 10 digits). For the detailed description of the motion segmentation, character segmentation, and character recognition phases, see [8].

The task of plate detection stage is to determine, in a single frame (image), the locations of license plates, called *plate candidates* in following. To this aim, we employ tree-based genetic programming, with each individual (program) representing a complete license plate detector. Each program is a tree composed of instructions (nodes) implementing various image analysis algorithms that pass

Table 1. Image processing instructions employed by the method

Operator	Arguments	Description
thr	image i, float t	Thresholds i using threshold t
gamma	image i, float γ	Gamma correction of i
inv	image i	Inversion of i
exp	image i	Pixel-wise exponentiation of i
log	image i	Pixel-wise logarithm of i
$+, -, *, /$	image i, image j	Pixel-wise image arithmetic

the processed images to each other. An input image is fed into programs using the terminal nodes (leaves). The image produced at the root node is interpreted as the output of a program, where pixel brightness is assumed to represent program's confidence in the presence of license plate at a particular location. This image is postprocessed using a fixed (i.e., non-evolving) procedure, which involves thresholding and scanning for connected components. Every connected component found is replaced by a corresponding minimal bounding rectangle (MBR). If an MBR fulfills certain size and aspect ratio constraints, the image fragment enclosed by it is passed to the character segmentation stage (and, subsequently, character recognition stage).

Table 1 presents the set of instructions used to form the evolving programs. The full set of instructions embraces also terminal nodes, which include: R, G, B (red/green/blue channel of the input image), and H, S, I (hue/saturation/intensity channels). Finally, the So terminal provides the input image converted to grayscale and processed using the Sobel filter, while F terminal is the output of simple handcrafted plate detector, which we obtained using evolutionary tuning in our previous study [8].

The algorithm evolves a population of programs encoded in the way described above. In each generation, a new population of programs is bred using selection, mutation, and crossover operators. Selection is driven by the fitness values assigned to programs. In the standard variant, the fitness of an individual program s (candidate solution) is defined as the performance of the complete recognition system that uses s (i.e., composed of: plate detection implemented by s, character segmentation, and character recognition using the trained SVM), averaged over the training set of images T. Formally,

$$f_{std}(s) = \frac{1}{|T|} \sum_{t \in T} f(s,t) \tag{1}$$

where $f(s,t) \in [0,1]$ is the performance of s on example (image) t, based on the agreement of the character sequence recognized at the plate location indicated in t by s, and the true character sequence present in the license plate in frame t (see experimental part). Alternatively, we employ another fitness assessment method detailed in the next section.

4 Implicit Fitness Sharing

To learn an image analysis algorithm that robustly detects license plates, the training set should be diversified, embracing images of different cars, taken in various lighting conditions, from different aspects, at various visibility, etc. In such a diversified sample, some plates can be expected to be easier to detect and recognize than others. The ability to solve (recognize) the harder examples (plates) should be particularly appreciated during the learning process. An individual that acquires such capability at some stage of evolution should have greater odds for survival, even if it happens to fail on some easier examples. Unfortunately, the standard definition of fitness values all examples equally. As a result, the individuals that exhibit such unique skills may have problems to pass the selection stage.

Implicit fitness sharing (IFS), introduced by Smith *et al.* [14] and further explored for genetic programming by McKay [11,12], is a technique designed to overcome this deficiency. It weighs the reward granted for solving each example according to its difficulty, which is assessed based on how hard it appears to the individuals in the current population. Formally, the fitness f_{ifs} of an individual (candidate solution) s is defined as:

$$f_{ifs}(s) = \frac{1}{|T|} \sum_{t \in T(s)} \frac{1}{n(t)} \tag{2}$$

where $T(s) \subseteq T$ is the set of examples solved by s, and $n(t)$ is the number of individuals that solve t. Thus, the total amount of reward that any example t can pass onto individuals in population amounts to 1.0, and that amount is shared equally between the individuals that solve it. When all individuals in population P solve t, $n(t) = |P|$, and they all receive the same, minimal reward. If, on the other hand, t is solved by only one individual in P, $n(t) = 1$ and such an individual (and only it) will be granted the maximal reward of 1.0.

The fitness function defined by IFS entices individuals in population to solve examples that appear particularly difficult for the current state of the search process. Individuals that exhibit such unique capabilities are highly rewarded, which increases their odds for survival, and makes propagation of their traits to next generations more likely. In this respect, IFS may be seen as a diversity maintenance technique. Notably, it is also a rudimentary form of coevolution, as the fitness granted to an individual depends on the performance of the other individuals in the population.

Standard IFS assumes that an individual either solves an example or not. In the license plate recognition task, the performance on a single example (image) may vary gradually, depending on the number of correctly recognized plate characters, and is reflected by the function $f(s,t) \in [0,1]$ (cf. standard fitness definition in Eq. (1)). To take this into account, we redefine the IFS fitness in the following way:

$$f_{ifs}(s) = \sum_{t \in T} \frac{f(s,t)}{\sum_{s' \in P} f(s',t)} \tag{3}$$

Contrary to the standard IFS (Eq. 2), in this formula the expression in denominator calculates the total performance of all individuals in population P (including s) on example t (rather than *counting* the number of individuals that solve t). Nevertheless, the effect is analogous: performing well on an example that is hard to recognize/classify is more beneficial than doing so for an example that is deemed easy by the individuals in population.

5 The Experiment

The primary objective of the experiment was to assess the performance of the proposed approach and verify the usefulness of implicit fitness sharing. Thus, the following configurations have been considered: conventional genetic programming (GP) with standard fitness function (Eq. 1) and genetic programming driven by IFS (GP-IFS, Eq. 3).

The Data. The image data is a part of collection of 1233 frames of 160 different vehicles (mostly passenger cars) described in our former study [8]. The training and testing sets are disjoint and comprise, respectively, 97 and 98 images randomly selected from that database. Each frame has been manually inspected and the actual (true) license number has been assigned to it. Frames have been acquired using a stationary camera working with resolution 1280×960 pixels, located at 15-20 meters from the passing-by cars. The motion segmentation phase typically crops the frames to dimensions comparable to VGA standard, in which vehicles occupy on average 75% of the frame area. Most frames feature cars in frontal view. The plates to be recognized have typically dimensions of 150×30 pixels, however, they are often far from rectangular due to perspective projection and vehicle's tilt and yaw. The dataset has been acquired in realistic conditions and is highly heterogeneous: it comprises various lighting conditions (different time of the day, including backlight as well as plates directly exposed to sunlight), different whether conditions (both sunny and cloudy days), and with license plates subject to dirt and mounted at different heights relative to road level.

The Setup. For both GP and GP-IFS, we run generational evolution algorithm with a population of 100 individuals for 100 generations. Other parameters are set as follows: tournament selection with tournament size 7, tree-swapping mutation applied with probability 0.9, subtree-replacing mutation applied with probability 0.1. Evolutionary runs are repeated 10 times to lower the variance of performance. For the remaining parameters, we use the defaults of the ECJ package [9] that the evolutionary part of our framework is based on. The image analysis component employs the OpenCV library [2] written in C++. Communication between modules is facilitated via exchange of XML files.

The performance of individual s on example t has been defined as:

$$f(s,t) = \frac{d_{max} - \min\{d_{max}, d(s(t), act(t))\}}{d_{max}} \tag{4}$$

where $s(t)$ is the character string representing the plate number as read by s, $act(t)$ is the actual plate number present in t, and d is the Levenshtein distance

Table 2. Fitness (f_{std}) of the best-of-run individuals. Averages and medians over 10 evolutionary runs. Best-on-training is the test-set performance of the best of best-of-run individuals.

	Training set		Testing set		
Setup	Average	Median	Average	Median	Best-on-training
GP	0.890±0.033	0.897	0.572±0.210	0.681	0.745
GP-IFS	0.914±0.006	0.917	0.682±0.131	0.713	0.767

metric, i.e., the minimal number of insertions, deletions, and substitutions required to transform one character sequence into another. For the set of plates considered here, we set $d_{max} = 5$, so the maximal distance that positively contributes to fitness is 4 (most plate numbers used here had 7 characters). If $d(s(t), act(t)) \geq 5$, an individual scores 0 for the frame. For instance, this is the case when no plate candidate has been detected in a frame. For conventional fitness function (GP), individual's fitness is a normalized sum of $f(s,t)$ (see Eq.(1)) over all training examples. For GP-IFS, the fitness is Eq. (3).

In our previous study, we found out that discrimination of characters '0' (zero) and 'O' is extremely difficult for the SVM classifier. In the typeface used in the considered license plates, these characters differ only in aspect ratio, which can be easily distorted by perspective projection. Because good discrimination of these decision classes is impossible without, e.g., syntactic rules, we fuse them and treat these characters exchangeably.

The Results. In Table 2 we present the fitness of the best-of-run individuals for each setup, and their performance on a test set. Because f_{std} and f_{ifs} cannot be compared directly, the best-of-run individuals have been assessed using standard fitness f_{std}. GP-IFS clearly outperforms standard GP, particularly on the test set, which suggests that the use of implicit fitness sharing lowers the risk of overfitting. We verified this additionally by calculating the Pearson correlation coefficient between the training-set and testing-set performance over the 10 runs. For GP, no significant correlation was observed (0.03), while for GP-IFS, that correlation was strong (0.81).

The average is an unbiased estimator of the expected performance of a method, and as such allows meaningful comparison. However, a pragmatic human designer of a plate recognition system would not care much about these estimates; rather than that, he would look for the best performing program. To simulate this attitude, for each method, from the 10 best-of-run individuals, we selected also the individual that attained the highest fitness, and evaluated it on the testing set. The last column of Table 2 reports the outcomes of that evaluation. Also in this case, the IFS-based approach fares better.

The theoretical upper bound of f_{std} is 1.0. This however does not necessarily mean that perfect performance can be attained using a specific plate reader, by which we mean here the character segmentation algorithm and character

Fig. 1. The output (right) of the best-of-run individual of one of the GP-IFS runs, when applied to the input image shown in left inset. Pixel brightness reflects individual's confidence in the presence of plate. The plate has been obfuscated due to privacy concerns.

recognizer together. In general, a plate reader cannot be guaranteed to correctly segment every plate and correctly recognize all segmented characters. In all recognition systems considered in this experiment, we use the same character segmentation algorithm and the same recognizer (an SVM classifier trained on a separate data set of character images). It is then justified to ask: what is the maximal fitness that can be attained by the entire system equipped with this plate reader, given a *perfect* plate detector?

We answer that question by applying the plate reader to actual plate locations in images, manually determined by a human expert. It turns out that for the training set, such a system does not work perfectly, attaining $f_{std} = 0.9196$. The data presented in Table 2 shows that GP-IFS is very close to this limit. As a matter of fact, two out of 10 runs of GP-IFS reach this performance. For the testing set, the system based on human labeling reaches $f_{std} = 0.8694$. This confrontation proves that the performance of recognition systems produced by both GP and GP-IFS is much closer to the realistic upper limit that it may appear when judging from the fitness definition alone.

Figure 1 presents exemplary results of the plate detection process carried out by the best-of-run individual of one of the IFS runs. Light-colored regions indicate the locations where the filter's belief in plate presence is higher. It may be observed that the detector correctly identifies the location of the plate, and does not get distracted by regions that have color characteristics similar to license plates (e.g., the light background behind the car).

Table 3 presents the distributions of Levenshtein distance for the training and test set for the GP-IFS approach. Most errors consist in single-character mistakes, which suggests that once the plate is correctly detected, the plate can be read flawlessly or with a low number of mistakes. For the testing set, the share of perfect recognitions drops on average to 55.7 percent, from 82.1 for the training set, but relying on the best of best-of-run individuals (last row of the table) leads to substantial improvements.

The median size of evolved image programs, measured in the number of tree nodes (leaves and nonterminal tree nodes) is 136. However, one of the very well performing best-of-run systems (training fitness 0.9175, testing fitness 0.7122)

Table 3. Distribution of Levenshtein distance d for GP-IFS (percents)

Levenshtein distance d	0	1	2	3	4	5	> 5
Average on training set	82.1	10.2	1.3	0.0	2.1	3.1	1.2
Average on testing set	55.7	11.9	3.2	2.6	1.0	2.1	23.5
Best-on-training on testing set	59.8	14.4	6.2	4.1	1.0	0.0	14.4

comprised mere 76 nodes. Though this number is still quite weighty, we hypothesize that most of the evolved systems can be substantially simplified without significant impact on fitness. This however, requires a separate investigation.

Analysis of IFS Weights. To verify the impact of example weighing realized by IFS, we collected additional statistics. In each generation, for each training image t, we logged $\sum_{s' \in P} f(s', t)$, i.e., the sum of individuals' performances on t. This quantity, called *solvability* in following, occurs in the denominator of Eq. (3), and its reciprocal determines the weight of an example. Thus, the easier an example appears to the individuals in the current population, the greater its solvability. In Fig. 2, we plot the average, minimum, and maximum of solvability, calculated over all 200 generations of an exemplary run, individually for each training example. The examples have been sorted ascendingly with respect to average solvability. As there are 100 individuals in population and the performance on a single example is normalized to $[0, 1]$ interval (Eq. 4), solvability cannot exceed 100.

Analysis of Fig. 2 allows us to draw several conclusions. It turns out that our training set contains eight examples, grouping on the left, that have never been correctly recognized by *any* individual throughout the run. Apparently, these examples turned out to be too difficult for the capabilities of learners. It is interesting to notice that they could have been safely discarded from the training set, as, by remaining constantly unsolved, they never contributed to fitness differences between individuals, so they had virtually no impact on the learning process.

Looking at the right-hand part of the graph, there are no examples that have been solved by all individuals in all generations, although there are a dozen or two of them for which this was true in at least one generation (see the *max* curve). Roughly speaking, of all 97 examples, around 60 rightmost can be judged as easy, with the average solvability of 87.0 or more. However, contrary to the left part of the graph, these examples definitely contributed to the search process, as in some generations they have been solved by no more than 20 individuals (see the *min* curve).

Finally, Fig. 2 suggests that the most valuable part of our training set comprises roughly 30 examples, with indices from 9 to about 38. These frames were moderately challenging to the learners, and probably contributed the most to fitness variation. Interestingly, these examples may be considered as an analog to the concept of *ideal evaluation set* coined in coevolutionary algorithms [4]. It may be hypothesized that recognition systems trained only on these examples could attain decent performance.

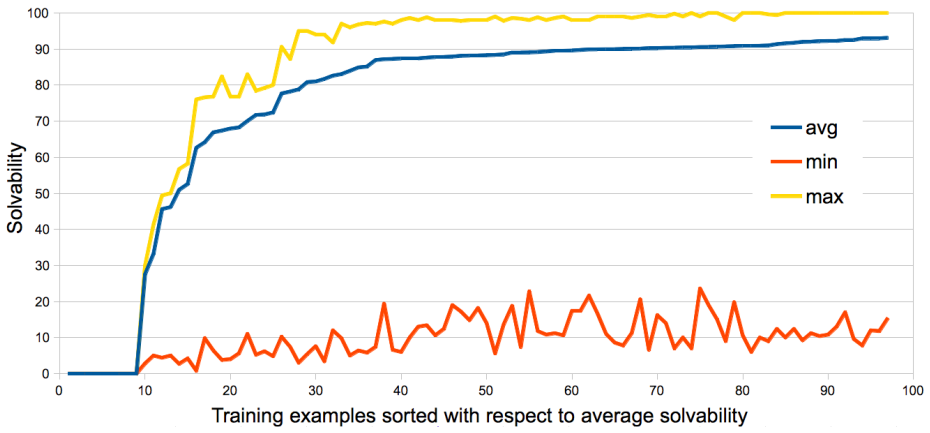


Fig. 2. Distribution of average, minimum, and maximum solvability of training examples, as estimated by IFS

6 Conclusion

The main conclusion of this study is that implicit fitness sharing is a useful alternative to standard evaluation when solving an object detection task. Because IFS is a general technique that abstracts from the nature of examples, it is justified to claim that some gains can be attained when evolving programs for other types of visual tasks, like image processing or object classification.

The recognition accuracy reported here is substantially greater than the one we obtained when using evolutionary algorithm to only *tune* the parameters of a fixed plate detection system [8]. On one hand, this result was expected, as the space of all possible image analysis programs is much greater than the space of parameters of a fixed image analysis program, and better (or at least not worse) performing programs can be found in such a bigger space. On the other hand, more expressive representation of solutions (complete programs vs. vectors of parameters) makes overfitting more likely, so running the above experiments was necessary to verify the test-set performance of plate recognition systems.

Typically, a passing-by car is captured in a few consecutive frames. By applying the approach reported above to multiple frames and aggregating the results, additional boost in recognition accuracy can be obtained. This is one of extensions that we intend to investigate in the follow-up of this study.

Acknowledgment. M. Nawrocki acknowledges support from European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°218086. K. Krawiec acknowledges support from National Science Center grant DEC-2011/01/B/ST6/07318.

References

1. Abdullah, S., Khalid, M., Yusof, R., Omar, K.: License plate recognition using multi-cluster and multilayer neural networks 1, 1818–1823 (2006)
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software, http://www.csie.ntu.edu.tw/~cjlin/libsvm
4. de Jong, E.D., Pollack, J.B.: Ideal Evaluation from Coevolution. Evolutionary Computation 12(2), 159–192 (2004)
5. For, W.K., Leman, K., Eng, H.L., Chew, B.F., Wan, K.W.: A multi-camera collaboration framework for real-time vehicle detection and license plate recognition on highways, 192–197 (June 2008)
6. Ji-yin, Z., Rui-rui, Z., Min, L., Yin, L.: License plate recognition based on genetic algorithm. In: 2008 International Conference on Computer Science and Software Engineering, vol. 1, pp. 965–968 (2008)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
8. Krawiec, K., Nawrocki, M.: Evolutionary Tuning of Compound Image Analysis Systems for Effective License Plate Recognition. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part I. LNCS, vol. 6922, pp. 203–212. Springer, Heidelberg (2011)
9. Luke, S.: ECJ evolutionary computation system (2002), http://cs.gmu.edu/~eclab/projects/ecj/
10. Martinsky, O.: Algorithmic and mathematical principles of automatic number plate recognition systems (2007)
11. McKay, R.I.B.: Committee learning of partial functions in fitness-shared genetic programming. In: 26th Annual Conference of the IEEE Third Asia-Pacific Conference on Simulated Evolution and Learning 2000, Industrial Electronics Society, IECON 2000, October 22-28, vol. 4, pp. 2861–2866. IEEE Press, Nagoya (2000), http://sc.snu.ac.kr/PAPERS/committee.pdf
12. McKay, R.I.B.: Fitness sharing in genetic programming. In: Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.G. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), July 10-12, pp. 435–442. Morgan Kaufmann, Las Vegas (2000), http://www.cs.bham.ac.uk/~wbl/biblio/gecco2000/GP256.pdf
13. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods – Support Vector Learning, MIT Press, Cambridge (1998)
14. Smith, R., Forrest, S., Perelson, A.: Searching for diverse, cooperative populations with genetic algorithms. Evolutionary Computation 1(2) (1993)
15. Sun, G., Zhang, C., Zou, W., Yu, G.: A new recognition method of vehicle license plate based on genetic neural network. In: 2010 the 5th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 1662–1666 (2010)

16. Tseng, P.C., Shiung, J.K., Huang, C.T., Guo, S.M., Hwang, W.S.: Adaptive car plate recognition in qos-aware security network. In: SSIRI 2008: Proceedings of the 2008 Second International Conference on Secure System Integration and Reliability Improvement, pp. 120–127. IEEE Computer Society, Washington, DC (2008)
17. Wang, F., Zhang, D., Man, L.: Comparison of immune and genetic algorithms for parameter optimization of plate color recognition. In: 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), vol. 1, pp. 94–98 (2010)
18. Zimic, N., Ficzko, J., Mraz, M., Virant, J.: The fuzzy logic approach to the car number plate locating problem. In: IASTED International Conference on Intelligent Information Systems, p. 227 (1997)

Feedback-Based Image Retrieval Using Probabilistic Hypergraph Ranking Augmented by Ant Colony Algorithm

Ling-Yan Pan and Yu-Bin Yang

State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing 210023, China
yangyubin@nju.edu.cn

Abstract. One fundamental issue in image retrieval is its lack of ability to take advantage of relationships among images and relevance feedback information. In this paper, we propose a novel feedback-based image retrieval technique using probabilistic hypergraph ranking augmented by ant colony algorithm, which aims at enhancing affinity between the related images by incorporating both semantic pheromone and low-level feature similarities. It can effectively integrate the high-order information of hypergraph and the feedback mechanism of ant colony algorithm. Extensive performance evaluations on two public datasets show that our new method significantly outperforms the traditional probabilistic hypergraph ranking on image retrieval tasks.

Keywords: Image retrieval, ant colony algorithm, semantic pheromone, hypergraph, feedback.

1 Introduction

With the explosive growth of digital images and other multimedia libraries, image retrieval has been actively studied in recent years [3,4,8,15]. Although image retrieval progressed rapidly with the current content-based indexing techniques, it has not yet succeeded in bridging the "semantic gap" between human concepts and low-level visual features [3].

A parallel line of research on this issue has highlighted several crucial points. The first problem is how to extract powerful descriptors to represent an image, e.g., bag-of-words image representation has shown effectiveness for image retrieval [14]. The second one is, how to explore the relationships among images and take advantage of the correlation information. Recent research literatures have remarkably focused on constructing vocabulary tree [15], integrating attributes [10], hierarchical indexing model [4], and simple graph-based learning. One common point of those methods is either using auxiliary textual information or considering merely pairwise relation.

Nevertheless, hypergraph [6,8,18] takes into account the relations among three or more vertices, which allows us to exploit the high-order information.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 387–396, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Bu et al. [2] used hypergraph to model the various objects and relations, and considered music recommendation as a ranking problem on this hypergraph. Gao et al. [6] presented the Hypergraph Laplacian Sparse coding (HLSc) and applied it to semi-auto image tagging problem. Following the algorithm in [18], Huang et al. [8] proposed a hypergraph based transductive algorithm for image retrieval. It tended to assign the same label to vertices that share many incidental hyperedges by assuming that predicted labels of feedback images should be similar to their initial labels. However, the incidence matrix of the above hypergraph is always simply assigned with binary values or relatively defined on the basis of the similarity between vertices, which is further applied to feedback-based image retrieval. Unfortunately, the similarity calculation is significantly sensitive to the distance measurements [13]. Besides, the incidence matrix further affects several parameters in the hypergraph. Thereby, an unreliable measure method induces a poor incidence matrix, which further influences hypergraph ranking results of image retrieval. Aiming at this problem, Li et al. [13] proposed a new affinity matrix generation method by using neighbor relation propagation principle.

In addition, combining relevance feedback information effectively to refine the search is still a considerably challenging problem. Fortunately, swarm intelligence (SI) [1], including ant colony optimization (ACO) [5] and particle swam optimization (PSO) [9] etc., is such a search strategy that exploits feedback mechanism with the help of insects. It is widely investigated in the last decades and can be used to solve a diversity of feedback-based learning tasks. Xue et al. [17] applied PSO to Feature Selection (FS) so that irrelative and redundant features are removed from a large feature space. Dorigo cooperated ant colony system to traveling salesman problem (TSP) [5]. Moreover, ants will mark favorable paths while foraging, which is similar to the way that users select positive images in retrieval. Specifically, for a query image, the first round of retrieval is carried without any extra information. Next, users will mark images that are more related and closer to the query one from the current related images, and then this feedback will benefits the next round. As can be seen, both of the marked paths and images are probably what the actors expect. In this paper, a novel feedback-based image indexing technique with probabilistic hypergraph ranking augmented by ant colony algorithm is proposed. It combines the

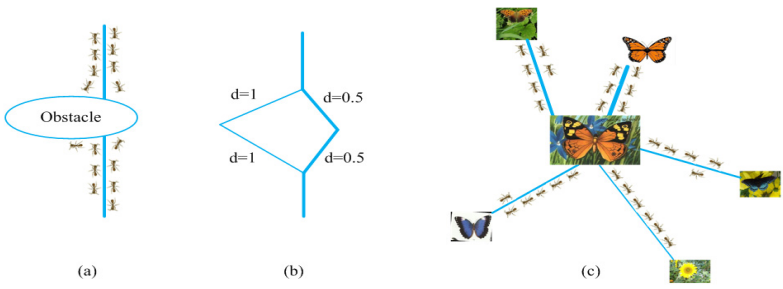


Fig. 1. The bolder line indicates denser pheromone concentration, and more related to the query image. (a) The real ants. (b) The artificial ants. (c) The ants' movement in image retrieval.

low-level features and semantic information to re-compute and enhance the affinity of the most related images, as well as increase their probability to be retrieved. It can effectively integrate the high-order information of hypergraph and the feedback mechanism of ant colony algorithm. Experiments show that the proposed method significantly improves the precision for image retrieval.

The rest of the paper is organized as follows. The related work is briefly introduced in Section 2. Section 3 describes the new approach that re-computing affinity matrix based on ant colony algorithm for feedback. In Section 4, we evaluate our algorithm on two standard datasets and demonstrate its effectiveness. Finally, conclusions and future work are provided in Section 5.

2 The Approach

2.1 Probabilistic Hypergraph

Denote V as the finite vertex set and E as the hyperedge set corresponding to a hypergraph of $G(V, E, w)$, where each hyperedge $e \in E$ is assigned with a positive weight $w(e)$ [18]. Let $A \in \mathbb{R}^{|V|\times|V|}$ represent the affinity matrix obtained with some measurement, where $a_{ij} \in A$ is normalized into $[0, 1]$, indicating the similarity between vertex v_i and vertex v_j. Then an incidence matrix can be defined based on matrix A:

$$h(v_i, e_j) = \begin{cases} a_{ij}, & \text{if } v_i \in e_j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Each element $h(v_i, e_j)$ in H indicates the probability of v_i existing in hyperedge e_j. Following [8], we can further compute the hyperedge weight $w(e)$, vertex degree $\delta(v)$ and hyperedge degree $\delta(e)$ based on the incidence matrix. Image retrieval on hypergraph can be considered as a ranking problem to minimize the sum of a normalization cost term $\Omega(f)$ and a regularization term [8,18]:

$$arg \min_{f \in \mathbb{R}^{|\mathbb{V}|}} \{\Omega(f) + \mu \|f - y\|^2\} \tag{2}$$

where the vector f is the image labels to be learned, y is initial labels and μ is the regularization parameter.

The above derivation shows that almost all of the parameters in the hypergraph learning task, including H, $w(e)$, $\delta(v)$ and $\delta(e)$, are obtained based on the affinity matrix A. However, the affinity evaluation significantly depends on the distance metrics. To address this problem, we introduce a powerful measurement and augment the relevance feedback process by ant colony algorithm, which can effectively search the most related images and enhance their affinity ability.

2.2 Similarity Measurement

In our paper, we adopt histogram intersection which measures the overlap between the images within histogram space to construct similarity matrix [16].

Experiments prove that histogram intersection is more effective than other similarity measures, e.g., chi-square distance used in [8], L1 distance, or L2 distance.

We utilize the bag-of-words model to represent images. First, SIFT feature descriptors [7] of 16×16 pixel patches computed over a grid with spacing of 6 pixels are densely extracted. Secondly, we create a 1024-bin codebook with k-means. Then, each image can be represented by a histogram with soft-assignment coding method [7] (any other available coding is possible). In order to capture location-related features of an object, three levels of spatial pyramids are also adopted. The similarity matrix can be finally computed as:

$$s_{ij} = \sum_{l=0}^{L} z^l \sum_{k=1}^{D^l} min(H_i(k), H_j(k)) \tag{3}$$

where $H_i(k)$ and $H_j(k)$ are two histogram features at the k-th bin, L is the level of spatial pyramids, z^l and D^l are the weighting parameter and the total dimensions at level l, respectively.

2.3 Hypergraph Ranking Augmented by Ant Colony Algorithm

Some ant species will deposit pheromone on the path while foraging to attract more ants. When they come to some obstacles, they are going to choose their paths marked by strong pheromone in probability. Due to the capability of perceiving pheromone, the ants tend to follow paths where pheromone concentration is higher after some time. In image retrieval, when pseudo-feedback information is introduced, the images which are labeled as positive images by users (from the whole retrieved result) should be assigned higher affinity and get higher probability to return as the top ranked retrieval results. Fig. 1 shows an example of real ants' foraging and artificial ant's movement in image retrieval.

Motivated by the above assumption, we expand ant colony algorithm to image retrieval. Before probabilistic hypergraph ranking, we improve the quality of the affinity matrix through applying the semantic pheromone to re-compute the probability that a vertex belongs to some hyperedge with ant colony algorithm. Let $\mathrm{T} \in \mathbb{R}^{|V|\times|V|}$ represent the pheromone matrix, where $|V|$ indicates the number of vertices (images) in a hypergraph $G(V, E, w)$. Firstly, we initial the pheromone concentration of any two vertices v_i and v_j as $\tau_{ij}^t = 0$, where the superscript t indicates the current feedback round and before a new search $t = 0$. In order to take into account pseudo-feedback, and ensure positive images get strong semantic pheromone, we update the pheromone matrix T in the $t + 1$ round as follows:

$$\tau_{ij}^{t+1} = \begin{cases} (1-\rho)\tau_{ij}^t + 1/|P_i|, & \text{if } v_j \in P_i \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where $\rho \in (0, 1]$ is an evaporation rate, P_i means the positive set of vertex v_i. Actually, Eq. (4) consists of two procedures: 1) Pheromone evaporation. It will firstly evaporate with a rate ρ before accumulating new pheromone, so the item $(1-\rho)\tau_{ij}^t$ computes the remaining pheromone. 2) Pheromone accumulation. We

take the reciprocal of the number of positive images as the new pheromone. The initial affinity matrix A computed by some measurement directly reflects the distances of low-level visual features among images. Here we handle it as heuristic information. Then the probability of an affinity between vertex v_i and vertex v_j which is chosen to be enhanced is

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[a_{ij}]^{\beta}}{\sum_{v_l \in P_i}[\tau_{il}]^{\alpha}[a_{il}]^{\beta}} \tag{5}$$

α and β in Eq. (5) are two parameters used to control the relative importance of the semantic pheromone and the heuristic information, i.e., the similarity of low-level features. If $\alpha = 0$, it just depends on the low-level features, which indicates that the most similar low-level features will be returned. This is a case of stochastic greedy mechanism. $\beta = 0$ is another extreme situation in which only the amplification factor of semantic pheromone plays a role. That is, the process merely adopts pheromone, rather than any bias taken from heuristic information. We will discuss this in Section 4.

By incorporating the initial affinity and the probabilistic choosing factor to enhance affinity of images, we reconstruct the affinity matrix as $\hat{A}$ in Eq. (6).

$$\hat{a}_{ij} = a_{ij} + \varpi_{ij} * p_{ij} \tag{6}$$

where ϖ_{ij} denotes the weight to be enhanced of the affinity that v_i belongs to e_j. Therefore the affinity a_{ij} is enhanced by weight ϖ_{ij} with a probability of p_{ij}. Finally, we normalize the new affinity matrix into $[0, 1]$ by subtracting the maximum. Specifically,

$$\hat{\hat{A}} = \exp(\hat{A} - \max(\hat{A})) \tag{7}$$

Empirically, the weights that is too samll in Eq. (6), e.g., $\varpi_{ij} \propto 0$, will make our algorithm not work because the right item of the plus symbol is so small that it will be omitted. Conversely, if $\varpi_{ij} \gg a_{ij}$, the reconstructed affinity matrix of Eq. (6) will contain few elements with considerable values, which further results in a new sparse affinity matrix with Eq. (7) because most of $\hat{a}_{ij}$ is far less than the $\max(\hat{a}_{ij})$. To address this problem, we propose an approach to determine the weight to be enhanced, which treats the weight of each affinity equally and computes the average of the factor $[\tau_{il}]^{\alpha}[a_{il}]^{\beta}$ of the positive set. Specifically, it uses the following criteria:

$$\varpi = \frac{\sum_{v_i \in V}\sum_{v_l \in P_i}[\tau_{il}]^{\alpha}[a_{il}]^{\beta}}{\sum_{v_i \in V}|P_i|} \tag{8}$$

Additionally, we add a constraint $b = \varpi/\bar{A} \leq \varpi_u$ to prevent from fluctuation, where ϖ_u is the upper bound and $\bar{A}$ indicates the average of the original affinity matrix. b significantly represents how much the weight is away from $\bar{A}$. If the constraint does not hold, we decrease it by a ratio of 0.1 recursively. We will further discuss it in Section 4. The procedure of re-computing the affinity matrix with ant colony approach is listed in Algorithm 1.

Algorithm 1. Re-computing the affinity matrix with ant colony approach

1: Compute similarity matrix S based on histogram intersection in Equation (3) and then normalize to generate affinity matrix A;
2: Update semantic pheromone concentration matrix T with Equation (4);
3: Compute the probability of an affinity chosen to be enhanced with Equation (5);
4: Determine the weight to be enhanced;
compute the initial weight ϖ in Equation (8);
while $b > \varpi_u$
$\varpi \leftarrow (1 - 0.1) * \varpi$; $b = \varpi / \bar{A}$;
end
$\varpi \leftarrow (\sum_{v_i \in V} |P_i|/|V|) * \varpi$;
5: Construct the new similarity matrix $\hat{\hat{A}}$ through Equation (6) and (7).

3 Experiments

3.1 Experimental Settings

In this part we demonstrate the effectiveness and efficiency of our proposed algorithm by conducting substantial experiments on two standard datasets: 15 class scene dataset [12] and Caltech-101 [11].

For the parameters α and β in Eq. (5), we compare and analyze the precisions and the recalls of different sets. The spatial pyramids factors are similar to the set in [8] as $1\times1, 2\times2, 1\times3$; and $z^l = \frac{1}{3}$ for $l = 0, 1, 2$. The optimal hyperedge size is computed from experimental data. Each query process employs the first round of pseudo-feedback which randomly marked 5 positive images (the query image is included) and 5 negative images. Actually, we handle this with several lines of codes which allow computers instead of users to select positive and negative images randomly and automatically. Then retrieve the expected results from the remaining images. Every experiment is repeated for 30 runs and the average precision and recall is finally reported.

3.2 Experiments on 15 Class Scene Dataset

The 15 class scene dataset consists of 4,485 scene images falling into 15 categories, including "forest", "mountain", "office", etc. The number of images associated with each class ranges from 200 to 400. We firstly conduct an in-depth analysis on this dataset.

Comparison under Different Similarity Measurements. In this experiment, we do not adopt ant colony algorithm for the fair comparison with Ref. [8]. We also implement the method in literature [8] by ourselves, but our work get optimal values at a hyperedge size of 120 rather than 40 in [8]. The detailed comparison results are shown in Fig. 2. As shown in Fig. 2, although we only employ dense SIFT descriptors, histogram intersection remarkably improves the

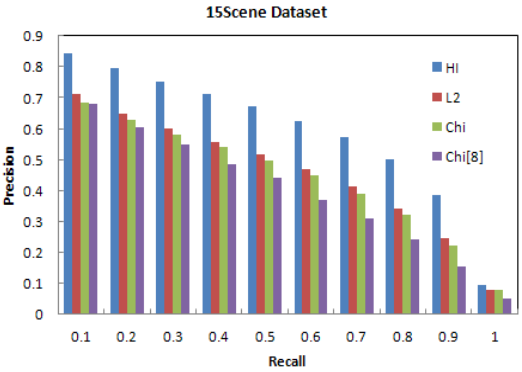


Fig. 2. The precision-recall curve for Scene dataset under different similarity measurements. *HI*: histogram intersection; *L2*: Euclidean distance; *Chi*: result implemented by ourselves with chi-square distance; *Chi[8]*: result in [8] with chi-square distance

performance by 2%∼13%. Hence, a powerful similarity measurement is of great importance in the probabilistic hypergraph ranking, which helps our work improve the quality of the similarity matrix.

Precision and Recall through Different Algorithms. As represented in Fig. 3(a), our method (green line) reconstructs the affinity matrix with ant colony algorithm and outperforms both the PHR in [8] (blue line) and the implementation by ourselves with histogram intersection (red line). APHR1 increases the precision from 84.49% of PHR-HI to 86.82% when recall is 0.1, and increases by approximately 7% from 50.48% to 57.29% when recall is 0.8. Therefore, our algorithm not only works well in low recall, but also available when recall goes higher. It is worth noting that the different sets of α and β according to APHR1 and APHR2. The case of $\alpha = 1, \beta = 1$ takes equal importance factor for similarity of low-level features and semantic pheromone. One extreme situation is $\alpha = 1, \beta = 0$ in APHR2 where only the semantic pheromone is in consideration when computing the enhancement of the affinity. Here we take the first feedback round for simple analysis. Then the reconstructed affinity matrix can be simplified as $\hat{A} = A + \varpi * \mathrm{T}$. T is always sparse with the elements corresponding to positive images are non-zero, herewith, only few values of A will be influenced by ϖ. If $\varpi_{ij} \gg a_{ij}$, the similarity of low-level descriptors is meaningless and has no influence for the retrieval. The corresponding affinity matrix is analogously computed by $\hat{A} = \varpi * \mathrm{T}$. As the positive images are assigned with the same label as the query one, A is converted to be a blocked diagonal matrix ordered by class label. Therefore $\hat{A} = diag(\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_{15})$, where $\hat{A}_i =$ is a $|V_i| \times |V_i|$ sparse matrix with only some elements are non-zero. $|V_i|$ is the total number of images labeled with the i-th class. The blocked diagonal matrix $\hat{A}$ makes the retrieval stuck within class and always gets rather higher but incredible precision, even 100%. We will discuss other sets of α and β in the following paragraphs.

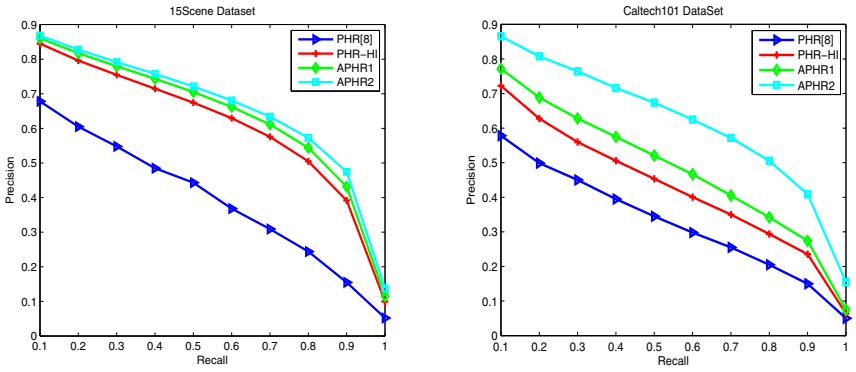


Fig. 3. Precision-recall results on different datasets. *PHR[8]*: Probabilistic Hypergraph Ranking in [8]; *PHR-HI*: our implementation of PHR which adopts the histogram intersection and only dense SIFT features; *APHR1*: PHR-HI with ant colony algorithm ($\alpha = 1, \beta = 1$); *APHR2*: PHR-HI with ant colony algorithm ($\alpha = 1, \beta = 0$)

Discussion. Parameter ϖ takes a significant place in our algorithm which is greatly influenced by α and β. Fig 4 shows the precisions with different sets of α when fixed β to be 1. As shown in the left sub-figure which computes the weight without constraint, when α raises from 0.2 to 4.0, the precision falls down from 87.16% to 84.59% for recall of 0.1. As both of the pheromone and the affinity are in $[0, 1]$, thus a large value of α will decrease the weight ϖ and further less improving the affinity matrix. Once $\alpha > 0.2$, ϖ is so small that it almost degenerates to traditional hypergraph ranking. However, the trend of growing is interrupted on $\alpha = 0$, because only the original affinity is in consideration in this case. It is worth noting that the precisions decrease to less than 30% for all recall when $\alpha < -0.5$, which is attributed to the large weight of ϖ. Specifically, if $\varpi_{ij} \gg a_{ij}$ and $\beta = 1$, the corresponding affinity matrix is analogously computed by $\hat{A} \approx \varpi * P$. The diagonal elements of P always have larger values than others, so a large enough ϖ will further results in a new affinity matrix with merely diagonal elements are 1 through normalization, which makes the performance rather poor. Therefore, a modification of ϖ is essential which leads to robust result as illustrated in right sub-figure of Fig. 4.

3.3 Experiments on Caltech-101 Dataset

Caltech-101 holds 9,144 images containing 101 object classes with high shape variability and a background class. The number of images per category varies from 31 to 800. Each class shares the same kind of object, but from changeable perspectives. The optimal hyperedge size is 40. The precision-recall curves are shown in Fig. 3(b). It illustrates that our implementation also outperforms the traditional hypergraph ranking which adopts chi-square distance metric by

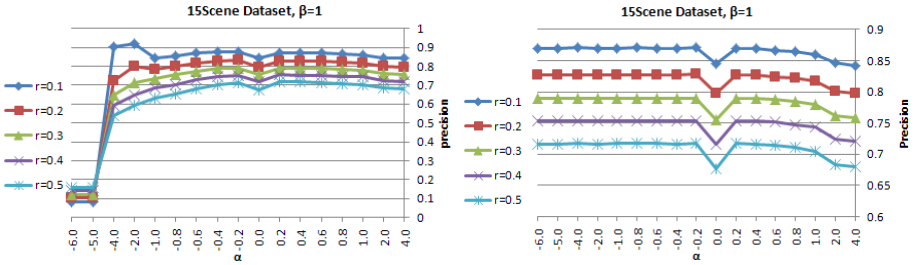


Fig. 4. The precisions under different α when $\beta = 1$, r indicates the recall. Left: no constraint on the weight ϖ; Right: have modified the weight.

2%-12%. The least improvement happens in the condition where recall is 1, while the precision raises from 57.8% to 77.2% when recall is 0.1. The performance is further advanced when taking an appropriated semantic pheromone with ant colony algorithm. However, a more impressive precision without any heuristic information is of less worth because of its local search.

4 Conclusion

In this paper, we propose a novel feedback-based image retrieval technique using probabilistic hypergraph augmented by ant colony algorithm, which aims at enhancing affinity between related images by incorporating both semantic pheromone and low-level features' similarity. It first constructs the similarity matrix based on low-level features with histogram intersection. Then it evaluates the probability of enhancing affinity with ant colony algorithm. Thus, the affinity matrix is re-computed to be applied to hypergraph learning task. Extensive experiments on two public datasets show that our new method significantly outperforms the traditional probabilistic hypergraph ranking.

However, currently the proposed method still has much room to be further investigated and improved. For example, we only take the first round feedback in our experiments, evaluation that a second or more rounds will further improve the performance is meaningful. We are also considering to apply ant colony algorithm to feature selection process as investigated by several literatures.

Acknowledgements. This work is supported by the Program for New Century Excellent Talents of MOE China (Grant No. NCET-11-0213), National 973 Program of China (Grant No. 2010CB327903), the Natural Science Foundation of China (Grant Nos. 61273257, 61021062, 61035003), and the Natural Science Foundation of Jiangsu, China (Grant No. BK2011005).

References

1. Bonabeau, E., Theraulaz, G., Dorigo, M.: Swarm Intelligence: From Natural to Artificial Systems, 1st edn. Oxford University Press, New York (1999)
2. Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., He, X.: Music recommendation by unified hypergraph: combining social media information and music content. In: Bimbo, A.D., Chang, S.F., Smeulders, A.W.M. (eds.) ACM Multimedia, pp. 391–400. ACM, Firenze (2010)
3. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40(2), 1–60 (2008)
4. Deng, J., Berg, A.C., Li, F.F.: Hierarchical semantic indexing for large scale image retrieval. In: CVPR, pp. 785–792. IEEE, CO., USA (2011)
5. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evolutionary Computation 1(1), 53–66 (1997)
6. Gao, S., Tsang, I., Chia, L.: Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. IEEE Trans. Pattern Anal. Mach. Intell. (2012)
7. van Gemert, J., Veenman, C.J., Smeulders, A.W.M., Geusebroek, J.M.: Visual word ambiguity. IEEE Trans. Pattern Anal. Mach. Intell. 32(7), 1271–1283 (2010)
8. Huang, Y., Liu, Q., Zhang, S., Metaxas, D.N.: Image retrieval via probabilistic hypergraph ranking. In: CVPR, pp. 3376–3383. IEEE, San Francisco (2010)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE, Perth (1995)
10. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: CVPR, pp. 951–958. IEEE, Miami (2009)
11. Li, F.F., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding 106(1), 59–70 (2007)
12. Li, F.F., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: CVPR (2), pp. 524–531. IEEE Computer Society Press, San Diego (2005)
13. Li, X.Y., Jie Guo, L.: Constructing affinity matrix in spectral clustering based on neighbor propagation. Neurocomputing 97, 125–130 (2012)
14. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV, pp. 1470–1477. IEEE Computer Society, Nice (2003)
15. Wang, X., Yang, M., Cour, T., Zhu, S., Yu, K., Han, T.X.: Contextual weighting for vocabulary tree based image retrieval. In: Metaxas, D.N., Quan, L., Sanfeliu, A., Gool, L.J.V. (eds.) ICCV, pp. 209–216. IEEE, Barcelona (2011)
16. Wu, J., Rehg, J.M.: Beyond the euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In: ICCV, pp. 630–637. IEEE, Kyoto (2009)
17. Xue, B., Zhang, M., Browne, W.N.: Multi-objective particle swarm optimisation (pso) for feature selection. In: Soule, T., Moore, J.H. (eds.) GECCO, pp. 81–88. ACM, Philadelphia (2012)
18. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) NIPS, pp. 1601–1608. MIT Press, Vancouver (2006)

An Evolutionary Approach for Automatic Seedpoint Setting in Brain Fiber Tracking

Tobias Pilic and Hendrik Richter

HTWK Leipzig University of Applied Sciences,
Faculty of Electrical Engineering and Information Technology,
Postfach 30 11 66, 04251 Leipzig, Germany
tobias.pilic@stud.htwk-leipzig.de, richter@eit.htwk-leipzig.de

Abstract. In this paper we present an evolutionary approach for optimising the seedpoint setting in brain fiber tracking. Our aim is to use Diffusion Tensor Imaging (DTI) data and Diffusion Magnetic Resonance Imaging (dMRI) data for feeding an automatic fiber tracking approach. Our work focusses on customising an evolutionary algorithm to find nerve fibers within diffusion data and allocate an appropriate number of seedpoints to them. This is necessary for the subsequent fiber reconstruction algorithms to work. The algorithm considerably enhances the speed and quality of the reconstruction and proves to be promising in leading to an automatic fiber tracking procedure used in medical imaging.

1 Introduction

Diffusion Tensor Imaging (DTI) or Diffusion Magnetic Resonance Imaging (dMRI) is an imaging method applied to measure the displacement of water molecules in human brain tissue [1][7]. Hence, it does not measure a geometrical structure but a physical process. This diffusion process is directly related to the tissue structure, which can be used to reconstruct nerve fiber structures within the brain. The scope of the work presented encompasses the implementation of an evolutionary approach for optimising fiber tracking in medical images of the human brain.

The result of DTI is a data set representing a three-dimensional picture in which every voxel contains a tensor. This diffusion tensor represents vector information for the evaluation of the main direction and strength of the diffusion process. The generation of this data is very time consuming. Therefore, it is desirable that a reconstruction process applied to the data is as fast as possible.

The current fiber tracking algorithms either cover the overall picture data or need an initial data set of seedpoints from which to start. The seedpoints are uniformly distributed within an area defined prior to image analysis, called room of interest (ROI). The standard procedure incorporates the manual adjustment of the seedpoints on the raw DTI data by medical or research staff. Although the DTI data can be plotted as a greyscale image, it is difficult to interpret and the ROI can partially miss the desired reconstruction area leading to a poor

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 397–406, 2013.
© Springer-Verlag Berlin Heidelberg 2013

brain fiber reconstruction [5]. Finding a sufficient ROI for every patient is a time consuming process as well.

DTI is an important method in neuroradiology and received much interest in research on physics, clinical applications and computational principles [2][5]. There are two constitutively different works which have used principles of evolutionary computation for fiber tracking. The algorithm GeneTrack [10] globally optimises fibers between two areas of the brain using a genetic algorithm. The second work [6] optimises the parameters of a reconstruction algorithm using global optimisation and tabu search. In this paper, we present an evolutionary approach which automatically optimises the positioning of the seedpoints within a given area. Thus, it overcomes the problem of having to manually adjust them [2][5]. Subsequently, we use FACT as a standard algorithm to perform brain fiber tracking [8].

First, we will briefly review the basic principles of Diffusion Tensor Imaging. Afterwards, the developed algorithm will be presented. Then, we will review the test environment for the algorithm. The final results will be discussed at the end of this paper.

2 Diffusion Tensor Imaging

2.1 The Physical Process of Diffusion and the Diffusion Tensor

Diffusion is a physical process occurring in liquids. It describes the behaviour of particles within a volume. Those particles are moving from places of high concentration to places of lower concentration to achieve a zero concentration gradient. The dynamic behaviour is described as follows [4]:

$$J = -D\nabla C. \tag{1}$$

Here, J represents the *Particle Density* (in $\frac{mol}{m^2 s}$) as a result of the *Concentration Gradient* ∇C. Due to the fact that diffusion is a spatial process affected by geometrical restrictions, the vectorial quantities J and ∇C do not necessarily point in the same direction. Therefore, the proportionality factor D extends to a $3x3$ semidefinite matrix, the *Diffusion Tensor*. The six scalar factors of this matrix are called *Diffusion Coefficients* (in $\frac{m^2}{s}$). They measure the ability of movement in a certain pair of directions. They can be interpreted as a geometrically related conductance. Hence, the *Diffusion Coefficients* are a measure of the underlying geometric structure.

Diffusion data is acquired by using special MRI sequences based on the spin-echo principle. A vast number of measurements is necessary for one set of data, thus, the regression process needed to calculate the *Diffusion Coefficients* is very time consuming.

2.2 Preprocessing and Interpretation of the Data

Prior to applying the diffusion data to the fiber tracking algorithm used in this paper, it has to be preprocessed.

Diffusion Ellipsoid. The *Diffusion Tensor* is reduced to three vectors representing an orthogonal coordinate system with three eigenvectors and three eigenvalues calculated from the tensor matrix D (1). An ellipsoid (Fig. 1 (a)) is used as the common form of representation for the resulting vectors as it is easier to interpret and visualize in DTI images.

Fractional Anisotropy. Anisotropy refers to the variation of diffusion in space and is especially related to the presence of particular tissues. *Fractional Anisotropy* is calculated as scalar value by using the eigenvalues λ_1, λ_2 and λ_3 from the *Diffusion Ellipsoid* representation [9]:

$$FA = \sqrt{\frac{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2}{2(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)}}. \tag{2}$$

This is the representation to be used in this paper as part of the fitness evaluation of our algorithm as it is easy to interpret and to handle. Fractional Anisotropy values range from 0 for isotropic diffusion to 1 for strong anisotropic diffusion.

Brain Tissue. The brain consists of two different types of brain tissues, the white and grey brain matter. Diffusion within the white brain tissue is highly anisotropic because of the extreme bundling of axons. Hence, this tissue consists of nerve fibers which are specialised on transferring data. Isotropic diffusion can be found within the grey brain tissue, where the neurons are entangled and diffusion has no preferred direction. The idea of brain fiber tracking is to identify areas of white brain tissue within the data and to use this data to calculate nerve tracts with the help of reconstruction algorithms. Therefore, the algorithm proposed in this paper will have to search for this type of tissue within the data. Places of white brain tissue characteristically have a *Fractional Anisotropy* value above 0.7 - 0.8. Thereby values ranging between 0 and 0.4 usually indicate grey matter or brain water.

Diffusion Dataset Utilised in This Work. A combination of the *Diffusion Ellipsoid* and *Fractional Anisotropy* reduced to a 2-D cross section of the original picture data is used in this work. This has shown to have no impact in terms of a different algorithmic behaviour while decreasing computation time. The vectors of the ellipsoid are projected to the cross sectional area and weighted with the voxel related value of the *Fractional Anisotropy*. This new form of data is shown in Fig. 1 (b).

2.3 Brain Fiber Tracking

There are two types of fiber tracking algorithms, the most commonly known ones are Streamline algorithms.

FACT Algorithm. One derivative of those Streamline algorithms reconstructs the fibers by simply following the main diffusion directions from a given seedpoint. It is called a FACT algorithm (*Fiber Assignment by Continuous Tracking* [8]). An example of such a reconstruction is shown in Fig. 1 (b), where the FACT is basically following the vectors from one voxel boundary to the next. Between every two consecutive voxels, the algorithm checks several termination criteria derived from clinical validations. The FACT method is the second step in evaluating the fitness within the algorithm developed in this paper because it is one of the simplest approaches for reconstructing brain fibers.

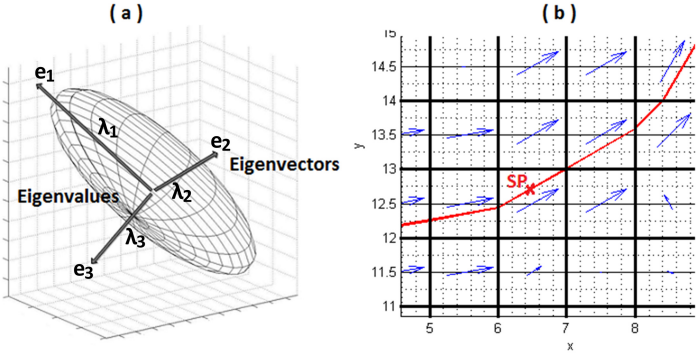


Fig. 1. Diffusion Ellipsoid and FACT Reconstruction. In Fig. 1 (a) the *Diffusion Ellipsoid* positioned within the laboratory grid of the MRI system can be seen. Notice that every voxel has a coordinate system different from this grid, which further complicates the handling of data. Therefore, the ellipsoid representation was chosen as a trade-off between easy handling and interpretability. Fig. 1 (b) shows a DTI picture with several pixels containing vectors (marked blue). They represent the 2-D reduction of the data used in this paper. The FACT reconstruction (marked red) starts at the seedpoint (SP) and follows the main diffusion directions given by the vectors.

Optimisation Problem. Our algorithm must position a set of seedpoints within a continuous search space. The optimisation problem is to find nerve fibers in this search space and to position the seedpoints in a manner that the fiber can be fully reconstructed by using Streamline algorithms.

The underlying idea of the algorithm is to find typical places of a high *Fractional Anisotropy* for nerve fibers and then evaluating the length of the fibers in those places to assess the quality of the position.

3 Evolutionary Approach

The individuals as members of the population are the seedpoints for the brain fiber reconstruction. Ideally, they should search the picture for fibers and position themselves at those fibers. It is intended that the overall cross-section of the fiber

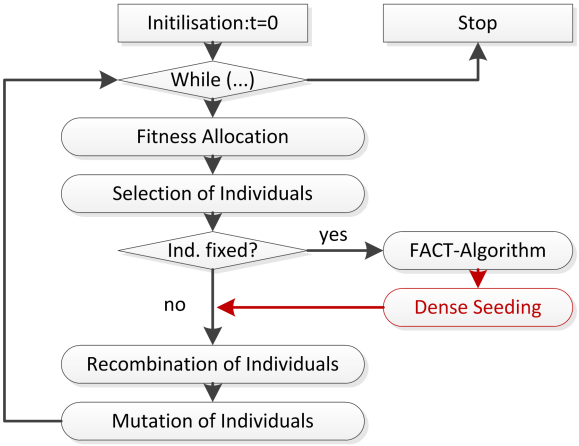


Fig. 2. Flow Chart of the Evolutionary Algorithm. The structure shown is typical for an evolutionary approach, except for the second loop, which was added to particularly assign the algorithm to the application. Thus, the process of dense seeding can be skipped, resulting in two possible and fundamentally different algorithms.

is sampled and longer fibers are favoured over shorter ones. A simple program flow is presented in Fig. 2. We iteratively added additional mechanisms and modified a simple *Evolutionary Algorithm* to solve our optimisation problem.

Initialisation. The individuals are initialised and uniformly distributed across the picture. The total number of the initial population is calculated by the number of pixels multiplied by a parameter, which is then handed over to the function (Table 1). Therefore it is adapting itself to the picture size.

It is unlikely that no individual immediately hits a fiber after initialisation. If this rare case comes into effect, the individuals, driven by mutation, are randomly moving around searching for fibers within the picture. If one or more individuals are getting fixed, other mechanisms such as recombination start to affect the movement of the individuals.

Fixation of Individuals. The individuals in a generic *Evolutionary Algorithm* applied to the diffusion data would keep on moving throughout the picture most of the time. But the individuals leading to a good reconstruction are intended to be kept and presented at the end of the algorithm. This process is called fixation. The fixed individuals do not undergo selection and recombination but still can be subject to mutation for maintaining the dynamic and probabilistic behaviour of the algorithm. Furthermore, fixed individuals have a better fitness if they enable a long fiber reconstruction and are passed onto the dense seeding function if available.

Fitness Evaluation. The *Fractional Anisotropy* FA (2) is assigned to the individuals as fitness value at the beginning of each iteration by searching the pixel in which the respective individual I is positioned. At a certain point during each

iteration of the main loop, the fixed individuals are separated and treated differently. The FACT algorithm is applied to them based on their current position. The fiber reconstruction resulting from the FACT algorithm is saved as a polygon $FACT(I)$. The length $\mathcal{L}(...)$ of the polygon is used to update the fitness.

$$fit = FA + \mathcal{L}(FACT(I)). \tag{3}$$

Individuals are rewarded for yielding to a very long fiber reconstruction. The fixed individuals are supposed to be moving on the fiber without loosing it. Because the mutation is fitness weighted, the resulting movement for the individuals is restricted to a quarter of a pixel. The reconstructed fibers are kept as recombination reference. It should be mentioned that the FACT parameters are fixed, therefore, they are not affected by the developed algorithm.

Selection, Recombination and Mutation. We have chosen the simple approach of tournament selection for our algorithm.

Further on, every non-fixed individual is recombined with the next shortest point to a reconstructed fiber. The recombination is done by positioning the individual in the middle of the line between both points. This method was chosen as it had produced the best results. Consequently, every non-fixed individual is forced to move in every iteration to constantly scan the picture. Notice that the individuals do not move to the next fixed seedpoint, which would result in a good fiber, but move the shortest way towards the fiber itself. One reason for choosing this method is the need to position the seedpoints along the cross-section of the fiber, which does not necessarily mean positioning them close to each other. The individuals are still able to cover the different parts of the picture while searching for fibers on their way. It is even more likely for them to find unknown branches of fiber or even new fibers.

A certain percentage of the randomly chosen population size is subject to Gaussian mutation with a characteristic variance and zero mean (in order to fit various picture sizes). Both the percentage of the population size and the variance are parameters adjusted offline (Table 1) The movement driven by maximum mutation is inversely related to the fitness.

Dense Seeding. In DTI, dense seeding refers to the fact that it is often necessary to place more than one seedpoint within each voxel to reach all branches of the fiber without loosing it at the edge. The lower the resolution of the picture, the more difficult it is to reach everything from a remote seedpoint area (ROI). An example of this problem is shown in Fig. 3 using 2-D test data.

To achieve dense seeding, the rooting of the individual with the highest fitness has to be permitted. A new number of individuals is normally distributed around it, depending on that fitness value. This dependency is called the growth of the seed. Immediately afterwards, the new seeds are checked for whether they meet the fixation criteria. If this is not the case, they are discarded, just as real seeds would die on infertile ground. The rest is kept and considered a direct result of the algorithm.

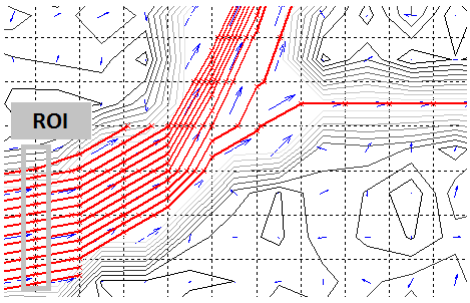


Fig. 3. Dense Seeding. The figure shows a small clipping of brain fiber including a branch. The position of the seedpoint across the fiber can be observed to have a severe impact on the resulting reconstruction. Only one seedpoint is able to reach the small branch and four other seedpoints are totally losing the fiber at the edge.

4 Experiments and Results

4.1 Test Environment

The first step prior to conducting the experiments was to find a test environment for the algorithm. It was chosen by doing research on normally occurring nerve fiber structures within the brain. Several sets of test data were designed with a size of $20x20$ pixel, containing examples of common geometrical structures with different complexities. This includes simple straight fibers, different widths, branches, intersections and singularities to test the proper termination behaviour of the fiber tracking algorithms. The white brain matter receives reasonable diffusion directions and *Fractional Anisotropy* values between 0.8 and 1. The gray brain matter consists of randomly generated directions with FA values between 0 and 0.4.

4.2 Parameter Optimisation and Results

Prior to development it was required to find an algorithm adapting itself to the various image sizes and types without being manually adjusted. Therefore, we started searching for a set of parameters to make the algorithm versatile while maintaining a high level of robustness. The total number of parameters was limited as much as possible to simplify the adjustment. Five different parameters appeared to be the best trade-off between adjustability and reduction (Table 1).

The algorithm was tested separately with and without the dense seeding functionality. In lack of an automatic evaluation procedure, the results were plotted, reviewed and assessed visually. The first step was to find a feasible range of values for the parameters. After that, the parameters were varied within those ranges, generating more than five thousand results. The next step was to scale down the range of values. The results were backed up by running a second simulation to eliminate uncertainties caused by the non-deterministic nature of the algorithm.

Table 1. Parameter Values

Parameters	Without DS	With DS
population size = fraction of image size	0.3	0.075
fixation value	0.3	0.3
mutation variance	1.5	1.5
mutation probability = percentage of population	0.45	0.45
termination value = percentage of fixed individuals	0.5	0.5
number of generations	2...10	2...5

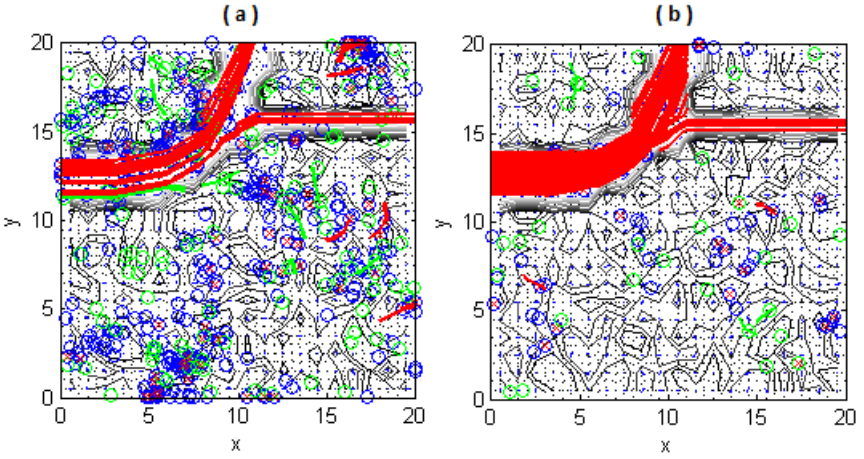


Fig. 4. Algorithm Results. The figures show the results of applying both algorithms to a relatively wide picture of low resolution with one fiber inheriting a small branch. The red marks and lines refer to the final result, the green to the starting population and the blue marks show the movement of the individuals during each iteration. Fig. (a) is the result of using the EA without dense seeding. The fiber can be discerned to have been fully touched. The small random fibers within the background of the picture were found as well. Fig. (b) refers to the EA with the dense seeding functionality. The fiber is fully reconstructed, concurrently using less iterations and individuals.

After several optimisation repetitions we found the parameters shown in Table 1. The results produced by the algorithm are presented in Fig. 4.

Algorithm Applied to Brain Slice. A final set of experiments was done with brain slices extracted from an example DTI data set [3]. The picture is bigger in size ($125x125$ pixel) in order to test the capability of the algorithm to work on different resolutions and a high number of small fibers. The results are presented in Fig. 5 showing both algorithms to be working well with this type of image.

4.3 Discussion

The results obtained with the parameters from Table 1 have been useful in solving the problems of the test environments described in section 4.1. The algorithm reaches all branches of nerve fibers (Fig. 4) because of the fiber weighted

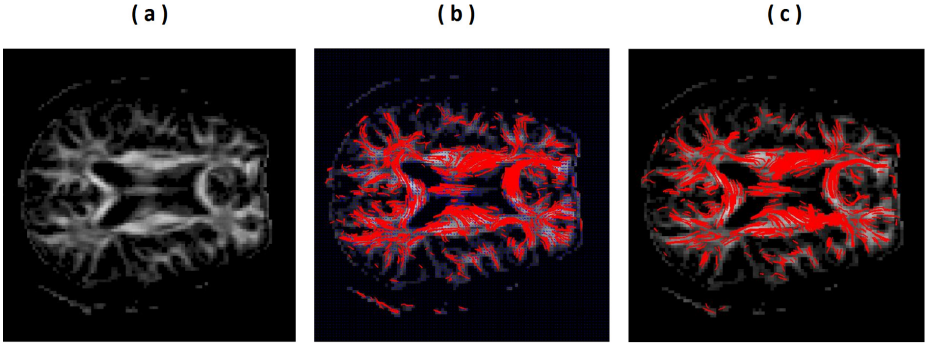


Fig. 5. Brain Slice Reconstruction. Fig. (a) shows the reference DTI data [3]. The second Fig. (b) is the result produced by the algorithm without dense seeding; Fig. (c) shows the algorithm with dense seeding.

recombination procedure. The procedure is encouraging individuals to find alternative ways to reach the fiber. The total number of individuals located in fibers is proportional to their length and width causing an appropriate distribution of the population across the image.

The evolutionary approach also works for images inheriting several fibers (Fig. 5) and adapts itself to images with varying characteristics and sizes. The mean number of generations is very small. Thus, computation time is decreased considerably by reducing the number of fitness evaluations. Essentially, the reason for this lies in the algorithm usually catching the first fibers immediately after initialisation. The search space is sufficiently covered by the individuals. Hence, no fibers were missed during test. Furthermore, the algorithm works autonomously, which could be developed into an automatic seedpoint setting procedure. The quality of the reconstruction was increased considerably when compared to a reconstruction where the seedpoints are uniformly distributed across the image.

There are big advantages in using the algorithm with the dense seeding function. The mean number of generations as well as the number of individuals are considerably smaller than they would be without DS (Table 1). On average, computation time is cut by half and the amount of data handled is further reduced. This means that the individuals are using less DTI data within the search space, which can be observed by comparing the blue traces within Fig. 4. There are also disadvantages to this method. The distribution of the individuals is not as smooth as it were without DS. Sometimes, the number of seedpoints is higher than necessary, in which case the parameters would have to be readjusted. Therefore, the algorithm with DS is most useful for fast seedpoint setting or as a primitive fiber reconstruction method. Without DS, the algorithm is perfect for a more robust seedpoint setting optimisation or for general nerve fiber detection.

5 Conclusion and Future Works

We have shown the prospect of using an evolutionary approach for automatically and optimally allocating seedpoints for brain fiber tracking in a ROI. The

seedpoints are positioning themselves and form the foundation of an automatic seedpoint setting procedure.

The first step in future works should be to expand the algorithm to 3-D, making it suitable for clinical applications. This would also enable the more substantiated comparison with state of the art algorithms and applications. At this stage, the evolutionary approach should also be combined with the various other types of fiber tracking algorithms [2] to evaluate the possibility of further enhancing it. It is also desirable to test the algorithm in combination with pattern recognition mechanisms to create a fully automatic ROI setting procedure. This procedure should be capable of navigating through DTI data and identifying its different regions. Embedding the procedure into the generic process of acquiring DTI data could considerably increase the clinical use of DTI images while cutting the high cost of MRI time.

The results of this paper alongside the results of previous works [6][10] are emphasising the beneftis of evolutionary approaches within DTI and in general medical imaging. They are useful replacements for generic mechanisms causing the original application to be faster, more efficient and robust.

References

1. Basser, P., Mattiello, J., LeBihan, D.: MR Diffusion Tensor Spectroscopy and Imaging. Biophys. J. V. 66 (1994)
2. Chung, H.-W., Chou, M.-C., Chen, C.-Y.: Principles and Limitations of Computational Algorithms in Clinical Diffusion Tensor MR Tractography. J. Neuroradiol. 32, 3–13 (2011)
3. Kroon, D.-J.: DTI and Fiber Tracking, `http://www.mathworks.com/matlabcentral/fileexchange/21130-dti-and-fiber-tracking`
4. Giancoli, D.C.: Physics for Scientists and Engineers, ch. 18. Prentice Hall (2000)
5. Hattlingen, E., Rathert, J., Jurcoane, A., Weidauer, S., Szelenyi, A., Ogrezeanu, G., Seifert, V., Zanella, F.E., Gasser, T.: A standardised evaluation of pre-surgical imaging of the corticospinal tract: where to place the seed ROI. Neurosurg. 32, 445–456 (2009)
6. Jose-Revuelta, L.M.S.: A Hybrid GA-TS Technique with Dynamic Operators and its Application to Channel Equalization and Fiber Tracking. In: Jaziri, W. (ed.) Tabu Search. InTech (2008)
7. LeBihan, D., Mangin, J.-F., Poupon, C., Clark, C., Pappata, S., Molko, N., Chabriat, H.: Diffusion Tensor Imaging: Concepts and Applications. J. of M.R.I. 13, 534–546 (2001)
8. Mori, S., Van Zijl, P.C.M.: Fiber Tracking: Principles and Strategies. NMR Biomed. 15, 468–480 (2002)
9. Mukherjee, P., Berman, J.I., Chung, S.W., Hess, C.P., Henry, R.G.: Diffusion Tensor MR Imaging and Fiber Tractography: Theoretic Underpinnings. AJNR 29, 632–641 (2008)
10. Wu, X., Xu, Q., Xu, L., Zhou, J., Anderson, A.W., Ding, Z.: Genetic White Matter Fiber Tractography with Global Optimization. J. Neurosci. Meth. 184, 375–379 (2009)

Prediction of Forest Aboveground Biomass: An Exercise on Avoiding Overfitting

Sara Silva[1,2], Vijay Ingalalli[1], Susana Vinga[1,3], João M.B. Carreiras[4] Joana B. Melo[4], Mauro Castelli[1,5], Leonardo Vanneschi[5,1], Ivo Gonçalves[2], and José Caldas[1]

[1] INESC-ID, IST, Universidade Técnica de Lisboa, 1000-029 Lisboa, Portugal
[2] CISUC, Universidade de Coimbra, 3030-290 Coimbra, Portugal
[3] FCM, Universidade Nova de Lisboa, 1169-056 Lisboa, Portugal
[4] Instituto de Investigação Científica Tropical, 1300-344 Lisboa, Portugal
[5] ISEGI, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal
sara@kdbio.inesc-id.pt

Abstract. Mapping and understanding the spatial distribution of forest aboveground biomass (AGB) is an important and challenging task. This paper describes an exercise of predicting the forest AGB of Guinea-Bissau, West Africa, using synthetic aperture radar data and measurements of tree size collected in field campaigns. Several methods were attempted, from linear regression to different variants and techniques of Genetic Programming (GP), including the cutting edge geometric semantic GP approach. The results were compared between each other in terms of root mean square error and correlation between predicted and expected values of AGB. None of the methods was able to produce a model that generalizes well to unseen data or significantly outperforms the model obtained by the state-of-the-art methodology, and the latter was also not better than a simple linear model. We conclude that the AGB prediction is a difficult problem, aggravated by the small size of the available data set.

1 Introduction

The importance of accurately estimating forest aboveground biomass (AGB) has been recognized in the literature (e.g. [13]). Forest AGB is a key component when assessing the carbon stocks of a given ecosystem, and mapping its distribution is paramount to monitor forests and capture deforestation processes, forest degradation, and the effects of conservation actions, sustainable management and enhancement of carbon stocks. Furthermore, it is a requirement of international conventions (e.g., United Nations Framework Convention on Climate Change, UNFCCC), especially on the basis of reporting mechanisms developed under the UNFCCC post-Kyoto Protocol and particularly the initiative focusing on Reducing Emissions from Deforestation and forest Degradation in developing countries (e.g. [1]). Remote sensing data acquired by sensors onboard orbital platforms provide the only means to assess and monitor the status and change

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 407–417, 2013.
© Springer-Verlag Berlin Heidelberg 2013

of biophysical characteristics of tropical forests in a global and systematic way. Numerous studies have demonstrated that a relationship exists between forest AGB and low frequency (L- and P-band) Synthetic Aperture Radar (SAR) data (e.g. [10]), though a high level of uncertainty still remains.

Genetic Programming (GP) is the automated learning of computer programs, using Darwinian selection and Mendelian genetics as sources of inspiration [15]. It is now a mature technique that routinely produces human-competitive results. However, a few open issues remain, overfitting being one of them. For a review of the state-of-the-art in avoiding overfitting in GP the reader is referred to [5]. The problem of AGB prediction has recently been used as a test case to assess the performance of an overfitting control technique, the RST [5]. The results showed a clear improvement when compared to the results of standard GP, but their quality was not assessed from the point of view of the application.

In this paper we tackle the AGB prediction problem using a much larger variety of methods. From classical regression methods, including recent improvements, to different variants and techniques of GP, including bagging and boosting, two GP techniques aimed at avoiding overfitting, and the cutting edge geometric semantic GP approach, they were all used in the context of a private "contest" that was launched with the goal of obtaining good models that can generalize well to unseen data. The next section describes the data and the terms of the contest. Section 3 describes the long list of methods used, while Section 4 reports and discusses the results. Finally, Section 5 draws some conclusions from this study.

2 Data

The dataset is composed of a combination of 112 forest AGB estimates and corresponding Advanced Land Observing Satellite (ALOS) Phased Array L-band Synthetic Aperture Radar (PALSAR) data covering the forested areas of Guinea-Bissau (West Africa). Forest AGB was estimated in two field campaigns that took place in 2007 (43 observations) and 2008 (69 observations). It was based on a stratified sampling methodology using an available land cover map of 2007. Individual trees were measured following a three-nest sampling plot methodology (4, 14, and 20m concentric sub-plots) and used in combination with allometric equations to obtain forest AGB estimates. ALOS PALSAR data was acquired in 2008 in fine beam dual (FBD) mode (i.e., HH and HV polarizations). After image processing, several metrics were extracted for the same locations (112 plots) that were measured in the two field campaigns. Those metrics were the minimum, maximum, mean, and standard deviation of the HH and HV polarizations, expressed in decibel (dB) units. Therefore we have eight features, that we designate as $x_1, ..., x_8$, where some are highly correlated, such as $x_2 - x_3 - x_4$ and $x_6 - x_7 - x_8$. More information about the data set can be found in [2].

For the contest, only 75 of the 112 samples were given to the participants, and the remaining 37 samples were held as the unseen data where to measure the quality of each proposed model. With the 75 samples the participants were free to do as they wished.

The extended data used in method 8 below (EXT-REAL) was obtained by randomly selecting 65536 samples from the study area (pixels from the image), with the only constraint that the distance between any two points cannot be lower than 200 meters. The synthetic extended data used in method 9 below (EXT-SYNT) was obtained by attributing to each of the eight features four different values ($4^8 = 65536$), equidistant from each other and inside the ranges given by the minimum and maximum values of each feature in the 75 samples.

3 Methods

A large part of the methods used and described next are based on GP, namely methods 5–13. All of these used 30 random partitions of the data (the 75 samples) as training (50 samples) and validation data[1] (25 samples). These partitions were the same for methods 5–11. The partitions were used as cross-validation to calculate the expected error, and in some cases to tune the parameters of the method.

Method 1 (LIN). The first method to be tested is multiple linear regression, using a stepwise selection algorithm, iterating forward selection and backward elimination steps based on the statistical significance of the regression coefficients. This procedure aims at having the simplest model. The model obtained uses only one feature: $y = 154.0373 + 8.7676x_6$. From now on it will be designated as LIN (linear).

Method 2 (LIN-NO). The second method is basically the same as LIN, but the model is fitted without the three detected severe outliers which have high Cook's distances. The model obtained is $y = 174.6253 + 9.8750x_6$ and it will be designated as LIN-NO (linear with no outliers).

Method 3 (EXP). Due to the asymmetry of variable V9, which has an exponential distribution with parameter 64.8839, and the non-normality of the errors obtained with the LIN and LIN-NO models, the logarithm transformation is tested. This resulted in the model $y = exp(8.1390 + 0.2680x_8)$, designated as EXP (exponential).

Method 4 (REG). The fourth method is standard linear regression with elastic net regularization [20]. The elastic net penalty is a linear combination of L1 and L2 regularization terms that aims at obtaining sparse weight parameters and assign similar weights to correlated predictors. The model obtained was $y = 191.6389 - 1.8963x_1 + 0.5056x_2 - 1.0050x_3 + 0.2156x_4 + 3.6368x_6 + 3.9242x_7 + 3.4831x_8$ and it will be designated as REG (regularization).

Method 5 (STD-GP). The fifth method is a common implementation of tree-based GP, using Dynamic Limits [17] for bloat control and a fixed maximum depth of 10. A population of 500 individuals, initialized with the Ramped Half-and-Half procedure [8], was allowed to evolve for 50 generations with standard

[1] In GP the validation data is often called test data. We call "unseen data" to the 37 samples that were not given to the participants, to avoid name confusion.

crossover and mutation (probabilities 0.9 and 0.1, respectively) and a replication rate of 0.1. The function set was composed of the four binary operators +, -, ×, and /, protected as in [8] and the terminal set included ephemeral random constants. Selection for reproduction was made with lexicographic tournament [11] of size 5. Elitism guaranteed the survival of the best individual into the next generation.

The resulting model (not shown) is, after simplification, a 67-node tree where features x_2, x_4 and x_7 do not appear, and it will be designated as STD-GP (standard GP).

Method 6 (WTD). The sixth method is similar to STD-GP, but it uses a weighted fitness function. The weighted fitness function is defined in terms of the Root Mean Square Error (RMSE):

$$f^* = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(W_i \cdot (E_i - P_i))^2} \tag{1}$$

where N is the number training samples, $\mathbf{W}$ is the weight vector, $\mathbf{E}$ is the expected values for the training data, and $\mathbf{P}$ is the predicted values for the training data. The weights are updated on every generation $\mathcal{G}$ with Algorithm 1. Fitness is given by $f = f^*$.

In the above algorithm, when $\eta = 1$, the values for $\mathbf{P^1}$ are already available. That is, the prediction values for generation $\eta = 1$ have already been calculated with $\mathbf{W} = \mathbf{1}$, so that we can update the weights for the next generation fitness function.

We update the weights depending on whether there has been any improvement in the prediction values over the generations. The magnitude of increase or decrease in the prediction values is reflected in the error value ε. The choice of error function ε is so that the updated weights do not reach saturation values for a small error differences. In other words, the error function $\varepsilon = 1 - (\varepsilon^\eta + 1)^{-1/2}$ is a slowly growing function in terms of differences between the expected and predicted values. If it were not the case (e.g., using exponential functions), then, even for a small deviation of predicted values from the expected values, we would have $\varepsilon \approx 1$, which should be avoided.

This approach of weighing is different form the usual weighing procedures (e.g. [14]), where each sample is re-weighed with respect to other samples in the

Algorithm 1. `Update Weights`

DEFINE: $\boldsymbol{\varepsilon}^\eta = |\mathbf{P}^\eta - \mathbf{E}|$ and $\boldsymbol{\varepsilon}^{\eta-1} = |\mathbf{P}^{\eta-\mathbf{1}} - \mathbf{E}|$ for any $\eta \in 1 \ldots \mathcal{G}$
$\boldsymbol{\varepsilon} = \mathbf{1} - (\boldsymbol{\varepsilon}^\eta + \mathbf{1})^{-1/2}$
INITIALIZATION: $\mathbf{P^0} = \infty$ and $\mathbf{W^0} = \mathbf{1}$
for $\eta \in 1 \ldots \mathcal{G}$ **do**
  $W_i^\eta = W_i^{\eta-1} * \varepsilon_i$; if $\varepsilon_i^\eta < \varepsilon_i^{\eta-1}$, for all $i \in 1 \ldots N$
  $W_i^\eta = W_i^{\eta-1} + (1 - W_i^{\eta-1}) * \varepsilon_i$; if $\varepsilon_i^\eta > \varepsilon_i^{\eta-1}$, for all $i \in 1 \ldots N$
  $W_i^\eta = W_i^{\eta-1}$; otherwise, for all $i \in 1 \ldots N$ RETURN:
  $\mathbf{W} = \mathbf{W}^\eta$;

training data. In this approach, re-weighing of a sample solely depends on the magnitude of change in error values and is not reflected upon by the magnitude of change of values of other samples.

The model that resulted from this method is, after simplification, a 17-node tree, which is very short for GP standards. We represent it with the expression $y = 2x_2 - x_3 + 3x_5 + x_6 + 73.078x_5/x_6$ and designate it as WTD (weighted).

Method 7 (WTD-17). The seventh method is basically the same as WTD, but uses the Dynamic Limits [17] with a fixed maximum depth of 17. The model that resulted from this method (not shown) is, after simplification, a 54-node tree where all features except x_2 appear, and it will be designated as WTD-17 (weighted with maximum depth 17).

Method 8 (EXT-REAL). This method is inspired by the work of Robilliard and Fonlupt [16]. Since their validation set was very small, they gathered thousands of additional samples from which the expected output was unknown, only knowing what reasonable bounds they should have. With this extended data set a new validation criterion was used: the lowest number of samples out of bounds, the better the model.

In this method we use real extended data as described in Section 2. When using the extended data D_{ext} of size N_{ext}, we make slight modifications to Equation (1). Let $UB = \max(\mathbf{E})$ be the maximum expected value in the training data, $LB = \min(\mathbf{E})$ the minimum expected value in the training data, and P_{ext} the prediction values for the extended data. We now define a "confidence" parameter as $c = \left| P_{bnd} \right| * 100/N_{ext}$, where $P_{bnd} = \{P^i_{ext} \in [LB, UB]\}_{i=1}^{N_{ext}}$. The confidence parameter c quantifies the proportion of our predictions that are in the range of expected values. We modify the fitness function as $f = f^*/c$.

The model that resulted from this method (not shown) is, after simplification, a 83-node tree where features x_2 and x_8 do not appear, and it will be designated as EXT-REAL (extended real data).

Method 9 (EXT-SYNT). This method is similar to EXT-REAL, but the extended data D_{ext} is synthetic data as described in Section 2. The model that resulted from this method (not shown) is, after simplification, a 93-node tree where all the features appear, and it will be designated as EXT-SYNT (extended synthetic data).

Method 10 (BAG). This method is a bagging of GP models. Instead of taking the training data and obtaining one model from it, we perform τ trials to obtain τ models. Each trial uses a training set that is formed by randomly drawing, with replacement, the same number of samples as the original training set (n=75). Then the output of the model is the median of the τ outputs for each instance. We used the median instead of the mean because of frequent surges observed in the prediction values. τ was set to 10.

By construction, the model that results from this method (not shown) is an ensemble of models, hence complex and difficult to interpret. We will designate it as BAG (bagging).

Method 11 (BOOST). In the normal weighted approach (WTD) we update the weights on every generation for a given instance of training data. In this method we perform τ trials for the training data and update a weighted distribution $\mathbf{D}$ for *each trial*. Under this method, each trial τ uses the same set of training samples, which are drawn at random without replacement at the beginning of trial 1. We adopt the commonly used $Ada - Boost$ (e.g. [7,14]) to update our distribution. Let us define $\mathbf{P}^{t-1}$ to be the best prediction values for the previous trial. Since $\mathbf{D}$ is updated at the end of each trial, we have $\mathbf{W}^{\eta}$ updates available from Algorithm 1. For each trial we use fitness $f = f^*$.

The boosting approach usually employs evaluating a final hypothesis / function based on τ functions, evaluated for each trial [14]. We follow a naive approach of selecting a function whose RMSE is the best among the τ evaluated functions. It has also been observed that such a best hypothesis is obtained from the t^{th} trial, where $t > \tau/2$. This confirms that there is a good chance of improvement by re-weighing over the trials, than just re-weighing over the generations, as followed in WTD and WTD-17 approaches. τ was set to 10.

By construction, the model that results from this method (not shown) is an ensemble of models, hence complex and difficult to interpret. We will designate it as BOOST (boosting).

Method 12 (RST). This method is the Random Sampling Technique (RST). The RST was originally used to improve the speed of a GP run [4], however in [9] it was used to reduce overfitting in a classification task in the context of software quality assessment. With the RST the training set is never used as a whole in the search process. Instead, at each generation, a random subset of the training data is chosen and evolution is performed taking into account the fitness of the solutions in this subset. This implies that only individuals that perform well on various different subsets will remain in the population. Recently, Gonçalves *et al.* [5] have proposed a more flexible approach to the RST, where the size of the random subset and how often it is changed are parameters of the algorithm. The authors tested their technique on real-life datasets and found the best results by using only one random sample in each generation. They also showed that the RST with these settings produces parsimonious models.

Algorithm 2. `Boosting`

1 INITIALIZATION: $\mathbf{D^1} = 1/N$
2 **for** $t \in 2 \dots \tau$ **do**
3 $\quad D_i^t = (D_i^{t-1})^{1-L_i}$, for all $i \in 1 \dots N$
4 $\quad$ where

$$L_i = \frac{\left|P_i^{t-1} - E_i\right|}{\max\left|\mathbf{P}^{t-1} - \mathbf{E}\right|}$$

5 $\quad$ UPDATE: $\mathbf{W}^{\eta}$ from Algorithm 1
6 $\quad$ NORMALIZE: $\mathbf{D^t}$
7 $\quad$ RETURN: $\mathbf{W} = \mathbf{D^t} * \mathbf{W}^{\eta}$;

We used RST with these settings. As for the regular GP parameters, the settings were similar to STD-GP except for these differences: the population was allowed to run for 100 generations, the tournament size was 2% of the population size, no random constants were in the function set, elitism guaranteed the survival of the best individual into the next generation, and no bloat control was used except for the fixed depth limit of 17. The model that resulted from this method is, after simplification, a 29-node tree represented by the expression $x_2 - 3x_1 + x_4 - 4x_5 + 8x_6 + x_7 - 3x_8 - x_4/(x_2 - x_1 + x_7)$, from now on designated as RST (Random Sampling Technique).

Method 13 (GS-GP). This method is a GP system that uses the geometric semantic genetic operators recently created by Moraglio *et al.* [12]. By semantics it is meant the behavior of a program once it is executed on a set of data or, more specifically, the set of outputs a program produces on the training data. The geometric semantic operators directly search the semantic space, and they have a number of theoretical advantages compared to the ones of standard GP systems. In particular, as proven in [12], they induce a unimodal fitness landscape on any problem consisting in finding the match between a set of input data and a set of known outputs (like for instance classification or regression). This should facilitate evolvability [6], making these problems potentially easier to solve for GP. The geometric semantic operators also have a major drawback: they always create offspring that are larger than their parents, causing an exponential growth of the individuals. However, with the development of a novel implementation [19] we were able to use them efficiently. This new GP system evolves the semantics of the individuals without explicitly building their syntax, freeing us from dealing with exponentially growing trees during the evolution. Only the best individual found must be explicitly built. For more details see [19].

A population of 200 individuals was allowed to evolve until 10000 fitness evaluations were completed, using similar settings to the RST method with a few differences: the tournament was regular (not lexicographic) and absolutely no bloat control was used. Both semantic operators were used, with a higher than normal mutation rate (0.5) since it was recognized that the geometric semantic mutation requires a higher rate for good exploration of the search space [19]. The mutation step of the geometric semantic mutation was 0.001 as in [12]. The model that results from this method (not shown) is a very large individual that we have not attempted to simplify. We will designate it as GS-GP (geometric semantic GP).

Method 14 (BAG-SGB). Stochastic Gradient Boosting (SGB) [3] typically uses a base learner (in our case, decision trees) and constructs additive regression models by sequentially fitting the chosen base learner to current "pseudo"-residuals by least squares at each iteration [3]. At these iterations, a simple base learner is built using a random sub-sample of the training data (without replacement), which has been shown to substantially improve the prediction accuracy and execution speed, and makes the approach resilient to overfitting [3]. The final model is a linear combination of each simple base learner, which can be seen as a regression model whereby each term is a tree. Furthermore, Suen *et al.* [18] have demonstrated that

building and combining (by averaging in the case of regression) several SGB models on bootstrap samples of the training data set performs significantly better than an unique SGB model, and concluded that it was accomplished by variance reduction. Therefore, instead of building a single SGB model, several SGB models fitted to bootstrap samples (with replacement) of the original training set (n=75) were built (BagSGB). In this study 25 bootstrap replicates were used to build a BagSGB model. For more details see [2].

By construction, the model that results from this method (not shown) is an ensemble of models, hence complex and difficult to interpret. We will designate it as BAG-SGB (bagging of stochastic gradient boosting).

4 Results and Discussion

Table 1 shows the results obtained by each method. We report the results in terms of root mean square error (RMSE) and correlation (CORR) between predicted and expected outputs. For the methods that used some kind of cross-validation, e.g. all the GP methods (that did 30 runs, each one with a different data partition - see Section 3), the expected error was calculated as the mean or median RMSE and CORR obtained in the 30 runs. We have decided to report both mean and median because the variability between runs was very high, and hence the median becomes a better estimate of the error. We report the error obtained on the unseen data, and when available we also report the error obtained in the training data.

None of the methods was able to produce a model that generalizes well on the unseen data. All RMSEs are high and accompanied by low (negative for all GP models) CORRs. The model with lowest RMSE and highest CORR is the one produced by the best state-of-the-art method, BAG-SGB, matched by the first two linear models, LIN and LIN-NO, and followed by REG. The non-linear models behaved much worse. Among the GP models, STD-GP and WTD-17 were the ones with higher RMSE on the unseen data, surprisingly followed by RST. None of the GP models was able to accurately estimate the error, with exceedingly high values in the mean expected RMSE, median expected RMSE always too optimistic, and expected CORR showing similar values between mean and median but completely failing to guess the negative values obtained on the unseen data. The only models providing similar values for the mean and median expected RMSE were BOOST and GS-GP, GS-GP being the less optimistic one. GS-GP is also the one achieving, by far, lower RMSE and higher CORR on the training data. With such results on the training data we could expect GS-GP to generalize worse, but in fact it is also the best GP model on the unseen data. This is explained by the geometric properties of its operators [19]. However, if we take into consideration also the simplicity and interpretability of the models, GS-GP cannot be considered the best of the GP models; among all the models BAG-SGB also cannot be considered the best. That award goes to the most simple linear model, LIN. It is noteworthy that BAG-SGB is reported to achieve RMSE 26.62 and CORR 0.95 when using the entire original data set of 112 samples [2], and with the 75 samples it is not better than a linear model.

Table 1. Results of the different models. Best of each column is marked in bold.

	Expected RMSE		Expected CORR		Training Data		Unseen Data	
Techniques	Mean	Median	Mean	Median	(Median Values) RMSE	CORR	RMSE	CORR
LIN	n/a	n/a	n/a	n/a	54.27	0.47	**74.50**	0.11
LIN-NO	n/a	n/a	n/a	n/a	55.00	0.47	75.88	0.11
EXP	n/a	n/a	n/a	n/a	58.07	0.46	87.03	0.03
REG	n/a	n/a	n/a	n/a	53.62	0.49	76.17	0.05
STD-GP	225.04	74.14	0.16	0.15	52.69	0.53	1253	-0.20
WTD	313.28	68.81	0.09	0.06	54.94	0.51	81.42	-0.02
WTD-17	8853	68.57	0.14	0.15	50.36	0.53	115.02	-0.10
EXT-REAL	505.03	68.97	0.00	0.08	56.57	0.45	82.30	-0.02
EXT-SYNT	86537	64.64	0.08	0.10	57.16	0.47	82.02	-0.28
BAG	94.46	62.09	**0.24**	**0.25**	52.51	0.61	81.24	-0.21
BOOST	**59.41**	**57.98**	0.22	0.22	61.92	0.33	80.91	-0.14
RST	86.25	66.23	0.03	0.07	51.55	0.58	88.55	-0.30
GS-GP	67.09	64.48	0.15	0.15	**44.12**	**0.74**	79.56	-0.03
BAG-SGB	n/a	n/a	n/a	n/a	n/a	n/a	75.07	**0.14**

5 Conclusions

We have performed an exercise on predicting the forest AGB of Guinea-Bissau, West Africa. In the context of a privately launched "contest", 14 methods were used but none was able to produce a model that generalizes well to unseen data. Even the best state-of-the-art method was not better than a simple linear model, despite the literature reporting much better results when using a larger data set. We conclude that the AGB prediction is a difficult problem, aggravated by the small size of our data set.

Acknowledgments. The presented work was partially supported by national funds through FCT under contract Pest-OE/EEI/LA0021/2011 and projects PTDC/EIA-CCO/103363/2008 and PTDC/EEI-CTP/2975/2012, Portugal. The Secretaria de Estado para o Ambiente e Desenvolvimento Durável (SEAD) of Guinea-Bissau and the Ministry of the Environment of Portugal funded and logistically supported the development of the Carboveg-GB project. This research was conducted under the agreement of the Japan Aerospace Exploration Agency's (JAXA) Kyoto and Carbon (K&C) Initiative. JAXA is particularly thanked for their provision of the ALOS PALSAR data.

References

1. Campbell, B.: Beyond Copenhagen: Redd plus, agriculture, adaptation strategies and poverty. Global Environmental Change-Human and Policy Dimensions 19(4), 397–399 (2009)
2. Carreiras, J., Vasconcelos, M., Lucas, R.: Understanding the relationship between aboveground biomass and ALOS PALSAR data in the forests of Guinea-Bissau (West Africa). Remote Sensing of Environment 121, 426–442 (2012)
3. Friedman, J.: Stochastic gradient boosting. Computational Statistics & Data Analysis 38(4), 367–378 (2002)
4. Gathercole, C., Ross, P.: Dynamic Training Subset Selection for Supervised Learning in Genetic Programming. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 312–321. Springer, Heidelberg (1994)
5. Gonçalves, I., Silva, S., Melo, J.B., Carreiras, J.M.B.: Random Sampling Technique for Overfitting Control in Genetic Programming. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds.) EuroGP 2012. LNCS, vol. 7244, pp. 218–229. Springer, Heidelberg (2012)
6. Gustafson, S., Vanneschi, L.: Crossover-based tree distance in genetic programming. IEEE Transactions on Evolutionary Computation 12(4), 506–524 (2008)
7. Iba, H.: Bagging, boosting, and bloating in genetic programming. In: Proceedings of GECCO 1999, vol. 2, pp. 1053–1060 (1999)
8. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
9. Liu, Y., Khoshgoftaar, T.: Reducing overfitting in genetic programming models for software quality classification. In: Proceedings of the Eighth IEEE Symposium on International High Assurance Systems Engineering, Tampa, Florida, USA, March 25-26, pp. 56–65 (2004)
10. Lucas, R., Armston, J., Fairfax, R., Fensham, R., Accad, A., Carreiras, J., Kelly, J., Bunting, P., Clewley, D., Bray, S., Metcalfe, D., Dwyer, J., Bowen, M., Eyre, T., Laidlaw, M.: An evaluation of the alos palsar l-band backscatter – above ground biomass relationship over Queensland, Australia. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 3(4), 576–593 (2010)
11. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: Proceedings of GECCO 2002, pp. 829–836. Morgan Kaufmann (2002)
12. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric Semantic Genetic Programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
13. Pan, Y., Birdsey, R., Fang, J., Houghton, R., Kauppi, P., Kurz, W., Phillips, O., Shvidenko, A., Lewis, S., Canadell, J., Ciais, P., Jackson, R., Pacala, S., McGuire, A., Piao, S., Rautiainen, A., Sitch, S., Hayes, D.: A large and persistent carbon sink in the world's forests. Science 333(6045), 988–993 (2011)
14. Paris, G., Robilliard, D., Fonlupt, C.: Applying Boosting Techniques to Genetic Programming. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 267–918. Springer, Heidelberg (2002)
15. Poli, R., Langdon, W.B., Mcphee, N.F.: A field guide to genetic programming (March 2008), `http://www.gp-field-guide.org.uk`
16. Robilliard, D., Fonlupt, C.: Backwarding: An Overfitting Control for Genetic Programming in a Remote Sensing Application. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 245–254. Springer, Heidelberg (2002)

17. 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)
18. Suen, Y.L., Melville, P., Mooney, R.J.: Combining Bias and Variance Reduction Techniques for Regression Trees. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 741–749. Springer, Heidelberg (2005)
19. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP applied to predicting pharmacokinetic parameters. In: Proceedings of EuroGP 2013, Springer (to appear, 2013)
20. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B 67, 301–320 (2005)

Human Action Recognition from Multi-Sensor Stream Data by Genetic Programming

Feng Xie, Andy Song, and Vic Ciesielski

RMIT University, Melbourne, VIC 3001, Australia
{feng.xie,andy.song,vic.ciesielski}@rmit.edu.au
http://www.rmit.edu.au/compsci

Abstract. This paper presents an approach to recognition of human actions such as sitting, standing, walking or running by analysing the data produced by the sensors of a smart phone. The data comes as streams of parallel time series from 21 sensors. We have used genetic programming to evolve detectors for a number of actions and compared the detection accuracy of the evolved detectors with detectors built from the classical machine learning methods including Decision Trees, Naïve Bayes, Nearest Neighbour and Support Vector Machines. The evolved detectors were considerably more accurate. We conclude that the proposed GP method can capture complex interaction of variables in parallel time series without using predefined features.

1 Introduction

The widespread penetration of smart phones with various sensors onboard has opened up opportunities for applications of human action recognition. Even a consumer phone can be easily turned into a sensor platform which can both constantly transmit and process sensor input about the person who carries the phone. Accurate recognition of the person's actions can enable a wide range of applications such as timing an activity, creating a sport profile and assisting living and healthcare [15].

The data stream produced by the sensors can be viewed as time series as each sensor generates a sequence of observations at regular intervals. Furthermore data are received from multiple channels such as accelerometer readings in x, y and z axes. Patterns of human locomotion such as walking and running, likely exist in these multi-channel time series. Since one sensor is not sufficient to make sense of a person's action, the ability of handling multiple data streams is crucial in this domain. Another difficulty in human action recognition is the lack of prior knowledge on the correlation between certain data patterns and human actions. As a result, manually constructing suitable models or features for different actions is not very feasible.

We present a Genetic Programming (GP) based method to address the aforementioned issues. More specially, our research questions are:

1. How can a suitable GP based methodology be established to evolve human action recognition programs which can handle raw input from multiple body sensors?

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 418–427, 2013.
© Springer-Verlag Berlin Heidelberg 2013

2. How can this method handle multi-class of human actions?
3. How does the GP method compare to conventional classification algorithms?

Action detection problems can be formulated in two different ways: (1) As binary problems, for example, is the person running or not, or (2) As multi-class problems, for example, is the person the person running, or walking or standing, or none of these. Binary problems are easier to formulate and accuracies are higher, however, multi-class problems are more important in practice. In this paper we address both formulations.

2 Related Work

There is a massive body of literature on time series analysis. Most of the work is focused on the prediction of future values. However, there has been some work on other aspects such as detecting shapes and patterns in time series [6]. Most of the work on time series has been on a single time series, there has been very limited work on multi-channel time series.

There are existing studies using GP in time series analysis. Most of these are for forecasting. Kaboudan [4], for example, used GP for housing price prediction. The researcher found that the results produced by GP were more reliable and logically acceptable than those from neural networks. Wagner and Michalewicz [14] proposed GP with adaptive windowing for forecasting in a dynamic environment. Song and Pinto [13] applied GP to a sequence of video frames for motion detection problems. Hetland et al. [2] combined a pattern matching chip and GP to discover temporal rules useful for prediction. Xie et al. [16] used a GP-based framework to detect events of interest in a time series with background noise. Although these works have a different goal to ours, they do show the potential of GP to find rules in time series data without much human intervention.

Our work is similar to time series classification problem [12]. However, they are eventually different. The data mining algorithms usually require time series stream to be segmented into non-overlapped, fixed-size vectors as inputs. The proposed method is capable to deal with overlaps and detect action patterns with any length within a maximum search window size.

GP is most naturally used for binary classification, a negative output denoting one class and a positive output the other. There have been a number of proposals for extending GP to multiple classes. Kishore et al. [7] models an n-class classification problem as n binary classification problems. One Genetic Programming Classifier Expression(GPCE) is trained for each class which can recognise its own class. A strength of association is calculated for each GPCE to address conflicts where one instance is claimed by two or more GPCEs. Muni et al. [9] reported a novel approach in which one GP tree consists of n subtrees, each representing one of the n classes. Loveard et al. [8] proposed five ways of multi-class classification in GP: Binary Decomposition, Static Range Selection, Dynamic Range Selection (DRS), Class Enumeration and Evidence Accumulation. Of these methods, DRS is proved to be the best, and this is the method

used in our work. In this approach each evolved individual carries with it a mapping of the output values to classes and the mapping is generated as part of the evolutionary process.

3 GP Representations

We use tree based genetic programming. The function set is shown in Table 1 which lists the parameters and return types for each function. In addition to the four basic arithmetic functions, there are three extra functions for multi-channel time series. Two of these, `Window` and `Multi-Channel`, take the special terminals such as `Operation` and `Temporal Index`, listed in the terminal set (Table 2). Others take double types as input. These are from the sensor streams or returned values of other functions. In Table 2, each `Channel` m terminal denotes one value from a channel. These functions and terminals are explained below.

Table 1. Function Set

Functions	Parameters	Return Type
$\{ +, -, *, / \}$	1. Double, 2. Double	Double
Window	1. Double, 2. Temporal Index, 3. Operation	Double
Temporal_Diff	1. Double	Double
Multi-Channel	1. Channel Index, 2. Multivariable Operation	Double

Table 2. Terminal Set

Terminal	Value	Return Type	For Function
Channel m	Current value at Channel m	Double	*Any*
Operation	AVG,STD,DIF,SKEWNESS	Integer	Window
Temporal Index	$[1, 2^{window\_size} - 1]$	Integer	Window
Channel Index	$[1, 2^{num\_of\_channels} - 1]$	Integer	Multi-Channel
Multivariable Operation	MED,AVG,STD,RANGE	Integer	Multi-Channel

Function Window. This function is responsible for taking a selection of points from a time series. It has three parameters. The first one is the input in double. The value changes as data is continuously fed in. To "remember" past values, this function has a memory list which keeps the most recent S readings as $t_0, t_1, ..., t_{S-1}$, from the earliest to current. S is set to 12 in this study. The second parameter selects data points from the memory list. It reads the return value from terminal `Temporal Index` which randomly generates an integer in the range 1 to $2^{12} - 1$. The binary equivalent of this number is then used to determine which data points in the memory list will be selected. For example, a value of 11 of "`temporal index`" will be converted to binary giving 000000001011. Points t_8, t_{10} and t_{11} are selected as the eighth, the tenth and the eleventh bits are `true`. A value of 4095 would result in all 12 points in the memory list being selected. This mapping mechanism enables flexible point selection inside a

window and consequently helps find the duration of an action.The third parameter randomly selects one of four operations: `AVG`, `STD`, `DIF` and `SKEWNESS` on the points selected by the second parameter `Temporal Index`. These operations correspond to the calculation of average, standard deviation, sum of absolute differences and skewness of the selected points respectively.

Function Temporal_Diff. Function `Temporal_Diff` captures temporal differences. It takes one value from the stream input as its parameter and returns the difference between the current value and the previous one. It is effectively a window of size 2 so it can be considered as a special case of function `Window` which applies the subtraction operation on the last two points in memory list.

Function Multiple-Channel. Functions `Window` and `Temporal_Diff` handle only one sequence of values, namely readings from one channel. They can not capture patterns occurring across multiple channels. Hence function `Multi-Channel` is introduced which has two parameters: `Channel Index` and `Multivariable Operation`. The first parameter works in a similar fashion to `Temporal Index` in function `Window`. The range of index is in [1,$2^M - 1$] (M is the total number of channels). The second parameter randomly selects one of four multiple-variable operations: MED, AVG, STD and RANGE which operate on the current values of the selected variables and return the median, the average, the standard derivation and the distance between the maximum and minimum of these values.

4 Multi-class Classification

In this work we have used two ways to implement multi-class classification: (1) A single multi-class classifier. (2) An ensemble classifier based on binary classifiers. Given a classification problem with a set of classes $C = \{c_1, c_2, ..., c_n\}$, for each class a classifier $Classifier_j$ is evolved to classify class c_j (positive) against all other classes (negative). In total, $n - 1$ classifiers are generated. When one instance is claimed by multiple classifiers as positive, the classifier with higher accuracy in training will win.

5 Experiments

The GP runtime settings in experiments for this study are fairly standard. The population size is 1000. The maximum and minimum tree sizes are 8 and 2 respectively. The crossover, mutation and elitism rates are 85%, 10% and 5%. The evolution stops at a maximum of 50 generations. Each run is repeated 10 times and best individual is used for testing.

Table 3. Number of Instances in the Three Data Sets

Data Set	Class	Training	Test
Synthetic Data 1	Positive	200	103
	Negative	199	96
Synthetic Data 2	Positive	193	110
	Negative	206	89
Human Action Data	Sitting	1697	311
	Standing	1074	1345
	Walking	950	892
	Running	819	549
	Others	85	51

5.1 Data Sets

Three sets of multi-channel streams were used for evaluation. The first two are synthetic, containing two and five channels of input respectively. They are to validate the aforementioned methodology. The third is human action data, containing input from 21 channels. Table 3 shows the number of instances in each class. They have been split into training set and test set.

Synthetic Data. Both of the two synthetic data sets are designed for binary classification purposes. In Synthetic Data 1, there are two variables of streaming data input. A positive is defined as the presence of significant simultaneous changes(> 0.5) in both variables. To generate this data, each variable was initialised randomly. The value at each time interval was also random. The probability of a noticeable change in one variable occurring is 0.7.

Synthetic Data 2 is more complex as it has five variables. If any two of five channels simultaneously changed by more than 5 over two consecutive time intervals, then that is a positive. Similarly to the first data set, for each channel the next point is randomly generated with a probability of change 0.3. As we understand the data sets, the suitable features for them would be the differences between two adjacent points on each channel.

Human Action Data. This real world data set was collected from the built-in accelerometer, gyro, and magnetometer sensors of an iPhone4. An application was developed to read triaxial physical movement measurements from inertial units at a frequency of 30Hz. In total, there are 21 measurements (channels) available for recording. This data set involved one subject and no cross-subject validation was included in this research. The subject was asked to put the iPhone 4 in the waist pocket and to perform four actions in an order, `Sitting`, `Standing`, `Walking` and `Running`. The duration of each action was arbitrary.

To label the ground truth, the subject go "GO" to mark the end point of the previous action and the starting point of a new one. The voice recording is synchronised with the sensor data recording. The "GO" command also occurred at the same time as the subject changed action. The class labels of the recorded data are relatively accurate.

Data for Comparisons. Using the above data sets, the proposed GP method was compared with four conventional classification methods: *J48* (Decision Trees) [11],*Naïve Bayes* [3] , *IB5* (Nearest Neighbours) [1] and *SVM* (Support Vector Machine) [5, 10]. However, these methods can not directly work on time series data as they have no built-in sliding window mechanism, so we manually segmented the three data sets. For each channel, a window of fixed-length of W_s is used to build an instance. The value W_s is set to 2 for the two synthetic data sets and 12 for action data. This is to ensure that these non-GP methods will receive the same amount of information as GP.

These methods treat one instance as one row of values. However in multi-channel streams, there are multiple rows in one instance, so we flattened them into one row just like representing a matrix in a one-dimensional array. These are the raw stream inputs. In addition, two feature sets Set A and Set B are constructed for conventional classifiers. Set A calculates the temporal difference, that is the difference between two consecutive points. Therefore, $(W_s - 1)$ features are extracted for one channel. Set B contains the averages and the standard deviations of points of each channel at one window position . The number of attributes in the data for conventional methods are shown in Table 4. Set B is only used on human action data.

Table 4. Data Converted for Conventional Methods

Data Set	Window size	No. of Attributes (Raw Input)	No. of Attributes (Features)
Synthetic Data 1	2	4	$2 \times (2-1) = 2$
Synthetic Data 2	2	10	$5 \times (2-1) = 5$
Human Action Data	12	252	Set A $= 21 \times (12-1) = 231$ Set B $= 21 \times 2 = 42$

5.2 Results

All the methods for comparison use default parameters from the WEKA package. IB5 was used because we found 5 nearest neighbour often gave the best results on these data sets. The other algorithms has been tuned to achieve their best results.

Binary Classification. Table 5 presents the average accuracies, true positive rates (TP) and true negative rates (TN) on each test data by various methods. Each experiment is a binary classification, so there are 6 rows of results as there are 4 actions in the human action data. When manually constructed features are not available, the performance of the conventional classifiers is very poor. In the case of sitting, *J48* and *Naïve Bayes* appear to have a good accuracy of 90.12%. However, this is deceptive as their true positive rates are zero. Effectively they recognized nothing. The high accuracy is merely due to the dominance of negative cases in the data set. Most of the other results are not much better. The only good result from non-GP methods is obtained by IB5 on running and SMO on sitting.

Table 5. Test Results - Conventional Methods vs. GP- both on Raw Input (%)

Data Set	J48	Naïve Bayes	IB5	SVM	GP
Synthetic Data 1	51.8 TP : 100.0 TN : 0	49.8 TP : 27.2 TN : 74.0	51.8 TP : 58.3 TN : 44.8	48.2 TP : 0 TN : 100.0	**100.0** TP : **100.0** TN : **100.0**
Synthetic Data 2	55.3 TP : 0 TN : 100.0	57.3 TP : 10.1 TN : 95.5	50.3 TP : 48.3 TN : 51.8	44.7 TP : 100.0 TN : 0	**100.0** TP : **100.0** TN : **100.0**
3. Sitting	90.1 TP : 0 TN : 100.0	90.1 TP : 0 TN : 100.0	40.2 TP : 0 TN : 44.6	99.6 TP : 100 TN : 99.5	**99.7** TP : **100.0** TN : **99.7**
4. Standing	57.0 TP : 0 TN : 99.6	57.3 TP : 0 TN : 100.0	57.3 TP : 0 TN : 100.0	57.0 TP : 0 TN : 99.6	**96.6** TP : **92.1** TN : **99.9**
5. Walking	76.3 TP : 21.7 TN : 97.9	74.4 TP : 9.9 TN : 100.0	85.7 TP : 52.9 TN : 98.6	81.2 TP : 52.6 TN : 92.5	**A:97.7** TP : **97.1** TN : **98.0**
6. Running	34.2 TP : 88.5 TN : 22.7	68.2 TP : 100.0 TN : 61.4	96.4 TP : 94.4 TN : 96.8	20.8 TP : 84.9 TN : 7.3	**99.5** TP : **98.0** TN : **99.9**

Table 6. Test Results - Conventional Methods on Features vs. GP on Raw Input (%)

Data Set	J48	Naïve Bayes	IB5	SVM	GP
Synthetic Data 1	100.0 TP : 100.0 TN : 100.0	100.0 TP : 100.0 TN : 100.0	100.0 TP : 100.0 TN : 100.0	100.0 TP : 100.0 TN : 100.0	**100.0** TP : **100.0** TN : **100.0**
Synthetic Data 2	98.0 TP : 97.8 TN : 98.2	87.9 TP : 100.0 TN : 78.2	100.0 TP : 100.0 TN : 100.0	100.0 TP :100.0 TN : 100.0	**100.0** TP : **100.0** TN : **100.0**
3. Sitting	76.9 TP : 25.4 TN : 82.6	60.8 TP : 100.0 TN : 56.3	63.1 TP : 89.4 TN : 60.2	90.1 TP : 0 TN : 100.0	**99.7** TP : **100.0** TN : **99.7**
4. Standing	72.9 TP : 54 TN : 87.0	85.2 TP : 89.3 TN : 82.1	64.4 TP : 21.1 TN : 96.6	57.3 TP : 0 TN : 100.0	**96.6** TP : **92.1** TN : **99.9**
5. Walking	90.4 TP : 74.3 TN : 96.8	40.5 TP : 78.6 TN : 25.4	93.8 TP : 84.8 TN : 97.4	71.7 TP : 0 TN : 100.0	**A:97.7** TP : **97.1** TN : **98.0**
6. Running	96.1 TP : 96.0 TN : 96.1	77.4 TP : 100.0 TN : 72.6	97.9 TP : 88.0 TN : 100.0	82.6 TP : 0 TN : 100.0	**99.5** TP : **98.0** TN : **99.9**

Table 6 presents the results from these conventional methods on Set A features, comparing with GP. This is a somewhat unfair comparison between conventional methods using temporal features and GP using raw data. The rightmost column in the table is for GP, which is consistently the best performer in every task. Although the conventional methods operate on temporal differences, they still can only achieve comparable results to GP on the synthetic data. In particular, the true positives of these methods are rather poor, for example SVM on all four actions, J48 on sitting and standing and IB5 on standing. Their performance on four human action recognition tasks is much worse than GP. It should be noted that by using Set A features SMO result in worse performance than raw data. Set B features help SMO to achieve better accuracy on running(98.6%) and on walking(85.3%). In case of sitting detection, the result is slightly worse than using raw data but is still reasonable with an accuracy of 98.7%. However, it still failed to recognise any standing action. This feature set B did not bring any benefit to other non-GP methods. The details are not presented due to the space constraints.

Multi-class Classification. Table 7 shows a comparison of these methods treating human action data a multi-class problem instead of a set of binary problems. For the conventional methods, we can see that the use of features is effective and different methods react to features differently. *J48*, *Naïve Bayes* and *IB5*, achieved better results on Set A features, while *SVM* benefits from Set B features. Nevertheless, their performance is much worse than that of GP on raw data which is 93.7%, almost 14% higher than the best from these non-GP methods (80%).

Table 7. Test Accuracies from Multi-class Classification(%)

	J48	Naïve Bayes	IB5	SVM	GP
Raw Input	20.7	29.0	33.1	35.6	
Set A: Temporal Diff	63.9	80.0	58.1	20.6	**93.7**
Set B: AVG and STD	28.3	62.5	40.1	50.2	

Table 8. Ensemble of Conventional Classifiers

	Sitting	Standing	Walking	Running	Accuracy
Raw Input	SVM	IB5	IB5	IB5	42.5%
Set A: Temporal Diff	J48	Naïve Bayes	IB5	IB5	78.7%
Set B: AVG and STD	SVM	Naïve Bayes	IB5	IB5	47.9%

Ensemble Approach for Multi-class Classification. From the above experiments, we selected the best binary classifiers generated by conventional methods on each action, either using features, or on raw inputs. They are listed in Table 8. All classifiers in one row are combined together to form an ensemble. The process is described in Section 4. When an instance is classified as positive by

multiple classifiers e.g. Sitting and Standing. The most accurate classifier will label the instance. If none of the classifiers recognize the instance, then it is marked as *Others*. The accuracy of each ensemble is presented on the rightmost cell on that row.

For GP, the best classifiers trained for each action (as shown in rightmost column in Tables 5 and 6) can also work together as an ensemble to perform multi-class classification. The accuracy was improved slightly to 94.5% compared to 93.7% achieved by a single GP classifier. Compared to the best result of each row in Table 7, we can see that the ensemble approach did not bring benefits to conventional methods, but marginally helped GP to be more accurate.

Table 9. Confusion Matrix for GP Classifier Ensemble: (Accuracy:94.5%)

	Sitting	Standing	Walking	Running	*Other*
Sitting	311	0	0	0	0
Standing	0	1238	15	0	92
Walking	0	0	862	4	26
Running	0	0	11	538	0
Other	9	0	16	0	26

Table 9 shows the confusion matrix of using an ensemble of the best evolved binary classifiers for the five-class problem. A major advantage of this ensemble is that an outcome of "None of the above" (*Other* in Table 9) is possible. While this is an error for the current task, it would be a perfectly good outcome if the person was lying down. In the current task we think that most of these errors come from transitions between states.

6 Conclusions and Future Work

The results of the above investigation have clearly demonstrated the advantages of the GP approach to human action recognition from multi-channel data stream. We conclude that with a proper function set and terminal set, GP can evolve multivariable time series pattern recognition programs to differentiate various human actions based on a collection of body sensor input. The methodology does not require manually designed time series features and can handle raw input, so it can be applied to scenarios where domain knowledge about the actions is not available. This method can handle multi-class human actions either by a direct multi-class approach or by an ensemble of binary classification programs. In comparison with conventional methods, the high accuracies of our GP method are evident. It outperforms these methods even when they operate on temporal features rather than on raw input.

In our future work, we will analyse evolved individuals to gain some insight into the evolved rules. Another future adaptation of this work is to take transitions between different actions into account.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning 6(1), 37–66 (1991)
2. Hetland, M.L., Sætrom, P.: Temporal rule discovery using genetic programming and specialized hardware. In: Proc. of the 4th Int. Conf. on Recent Advances in Soft Computing, pp. 182–188 (2002)
3. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI 1995, pp. 338–345. Morgan Kaufmann Publishers Inc., San Francisco (1995)
4. Kaboudan, M.: Spatiotemporal forecasting of housing prices by use of genetic programming. In: The 16th Annual Meeting of the Association of Global Business (2004)
5. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to platt's smo algorithm for svm classifier design. Neural Comput. 13(3), 637–649 (2001)
6. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM 2005, pp. 226–233. IEEE Computer Society, Washington, DC (2005)
7. Kishore, J.K., Patnaik, L.M., Mani, V., Agrawal, V.K.: Application of genetic programming for multicategory pattern classification. Trans. Evol. Comp. 4(3), 242–258 (2000)
8. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 1070–1077. IEEE (2001)
9. Muni, D.P., Pal, N.R., Das, J.: A novel approach to design classifiers using genetic programming. Trans. Evol. Comp. 8(2), 183–196 (2004)
10. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods, pp. 185–208. MIT Press, Cambridge (1999)
11. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
12. Ratanamahatana, C., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining time series data. In: Data Mining and Knowledge Discovery Handbook, pp. 1049–1077 (2010)
13. Song, A., Pinto, B.: Study of gp representations for motion detection with unstable background. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2010)
14. Wagner, N., Michalewicz, Z.: An analysis of adaptive windowing for time series forecasting in dynamic environments: further tests of the dyfor gp model. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO 2008, pp. 1657–1664. ACM, New York (2008)
15. Wang, L., Gu, T., Tao, X., Chen, H., Lu, J.: Recognizing multi-user activities using wearable sensors in a smart home. Pervasive Mob. Comput. 7(3), 287–298 (2011)
16. Xie, F., Song, A., Ciesielski, V.: Event detection in time series by genetic programming. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (June 2012)

Novel Initialisation and Updating Mechanisms in PSO for Feature Selection in Classification

Bing Xue, Mengjie Zhang, and Will N. Browne

School of Engineering and Computer Science
Victoria University of Wellington, Wellington, New Zealand
{Bing.Xue,Mengjie.Zhang,Will.Browne}@ecs.vuw.ac.nz

Abstract. In classification, feature selection is an important, but difficult problem. Particle swarm optimisation (PSO) is an efficient evolutionary computation technique. However, the traditional personal best and global best updating mechanism in PSO limits its performance for feature selection and the potential of PSO for feature selection has not been fully investigated. This paper proposes a new initialisation strategy and a new personal best and global best updating mechanism in PSO to develop a novel feature selection algorithm with the goals of minimising the number of features, maximising the classification performance and simultaneously reducing the computational time. The proposed algorithm is compared with two traditional feature selection methods, a PSO based method with the goal of only maximising the classification performance, and a PSO based two-stage algorithm considering both the number of features and the classification performance. Experiments on eight benchmark datasets show that the proposed algorithm can automatically evolve a feature subset with a smaller number of features and higher classification performance than using all features. The proposed algorithm achieves significantly better classification performance than the two traditional methods. The proposed algorithm also outperforms the two PSO based feature selection algorithms in terms of the classification performance, the number of features and the computational cost.

Keywords: Particle Swarm Optimisation, Feature Selection, Classification.

1 Introduction

Classification problems usually have a large number of features, including relevant, irrelevant and redundant features. However, irrelevant and redundant features may reduce the classification performance due to the large search space, known as "the curse of dimensionality" [3,6]. Feature selection is to select a subset of relevant features for classification, which could shorten the training time, simplify the learned classifiers, and/or improve the classification accuracy [6].

Feature selection is a difficult problem due mainly to the large search space, which increases exponentially with respect to the number of available features [6]. Therefore, an exhaustive search is practically impossible in most situations.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 428–438, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Different heuristic search techniques have been applied to feature selection, such as greedy search [3]. However, most of the existing algorithms still suffer from the problems of stagnation in local optima or being computationally expensive [3,6]. In order to better address feature selection problems, an efficient global search technique is needed.

Evolutionary computation (EC) techniques are well-known for their global search ability. They have been applied to feature selection problems, such as genetic algorithms (GAs) [1], genetic programming (GP) [13], and particle swarm optimisation (PSO) [15]. PSO [14] is a relatively recent EC technique, which is computationally less expensive than some other EC algorithms. In PSO [14], a population of candidate solutions are encoded as particles in the search space. PSO starts with the random initialisation of a population of particles. Based on the best experience of one particle (*pbest*) and its neighbouring particles (*gbest*), PSO searches for the optimal solution by updating the velocity and the position of each particle according to the following equations:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{1}$$

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (p_{id} - x_{id}^{t}) + c_2 * r_{2i} * (p_{gd} - x_{id}^{t}) \tag{2}$$

where x and v represent the position and the velocity. t denotes the tth iteration in the evolutionary process. $d \in D$ denotes the dth dimension in the D-dimensional search space. w is inertia weight. c_1 and c_2 are acceleration constants. r_{1i} and r_{2i} are random values uniformly distributed in [0, 1]. p_{id} and p_{gd} represent the elements of *pbest* and *gbest* in the dth dimension.

Many studies have shown that PSO is an efficient search technique for feature selection [2,8,15]. However, there are some limitations about current PSO for feature selection. Firstly, PSO has not been tuned to the feature selection task. Many initialisation strategies have been proposed in PSO to improve its performance [16]. However, no existing initialisation strategies are specifically proposed for feature selection. Secondly, the traditional *pbest* and *gbest* updating mechanism may cause missing good feature subsets with high classification performance and a small number of features (Discussions in Section 2.2). Therefore, the potential of PSO for feature selection has not been fully investigated.

1.1 Goals

The overall goal of this paper is to propose a new PSO based feature selection approach to selecting a smaller number of features and achieving similar or even better classification performance than using all features and traditional/existing feature selection methods. To achieve this goal, we propose a new initialisation strategy and a new mechanism for updating *pbest* and *gbest* in PSO to reduce the number of features without reducing (or even increasing) the classification performance. Specifically, we will:

- propose a new initialisation strategy in PSO to reduce the number of features without decreasing the classification performance of the evolved subset,

- develop a new updating mechanism to lead PSO to search for the feature subsets with high classification performance and small numbers of features,
- develop a new PSO based wrapper feature selection algorithm using the proposed initialisation strategy and updating mechanism, and
- investigate whether the proposed feature selection algorithm can outperform two traditional feature selection methods, a PSO based algorithm with the goal of only maximising the classification performance, and a PSO based two-stage algorithm considering both of the two main objectives.

2 Proposed Approach

Feature selection has the two main objectives of maximising the classification performance and minimising the number of features. However, most existing methods only aim to maximise the classification performance [2]. Some works combine these two objectives into a single fitness function [8,18], but they need a predefined parameter to balance these two components, which is usually problem-dependent and hard to determine *a priori*. To solve this problem, we only include the classification error rate in the fitness function (Equation 3) because it is more important than the number of features. Meanwhile, we propose an initialisation strategy and a new *pbest* and *gbest* updating mechanism to reduce the number of features without decreasing or even increasing the classification performance, which also reduces the computational cost.

$$Fitness_1 = ErrorRate \tag{3}$$

2.1 New Initialisation Strategy

The new initialisation strategy is motivated by the two traditional methods, forward selection [17] and backward selection [12]. Forward selection starts with an empty set of features and it usually selects a smaller number of features, but it may miss the optimal feature subset with a large number of features. Backward selection starts with the full set of features. It usually selects a large number of features and the computational time is longer than forward selection.

Hence, we propose a new initialisation strategy to take the advantages of forward and backward selection and avoid their disadvantages. In this new strategy, particles are initialised using a small number of features. Therefore, the algorithm will start with searching the solution space with small feature subsets. This will also reduce the computational cost because the evaluation of a small feature subset in wrapper approaches takes less time than a large feature subset. However, if all the particles are initialised with small subsets, PSO may miss the medium or large feature subsets that can achieve the best classification performance. Therefore, in the proposed initialisation strategy, most particles are initialised using a small number of features (simulating forward selection) and other particles are initialised using large feature subsets (simulating backward selection). Meanwhile, through social interaction (updating *pbest* and *gbest*), PSO is expected to be able to reach and search the solution space with medium feature subsets if these feature subsets can achieve better classification performance.

2.2 New *pbest* and *gbest* Updating Mechanism

In PSO, particles share information through *pbest* and *gbest*, which can influence the behaviour of the swarm during the evolutionary process. Traditionally, the *pbest* and *gbest* are updated solely based on the fitness value of the particles (i.e., classification performance in feature selection problems). *pbest* of a particle is updated only when the fitness of the new position of the particle is better than the current *pbest*. In feature selection, the traditional updating mechanism has a potential limitation. If the classification performance of the particle's new position is the same as the current *pbest*, but the number of features is smaller, the particle's new position corresponds to a better feature subset. However, according to the traditional updating mechanism, the *pbest* will not be updated because their classification performance is the same.

To overcome this limitation, we propose a new *pbest* and *gbest* updating mechanism. In the new mechanism, the classification performance of the feature subset is used as the fitness function, which means the classification performance is still the first priority, but the number of features is also considered. *pbest* and *gbest* are updated in two situations. The first situation is that if the classification performance of the particle's new position is better than *pbest*, *pbest* will be updated and replaced by the new position. In this case, the number of features selected will be ignored. The second situation is that if the classification performance of the new position is the same as *pbest* and the number of features is smaller, the current *pbest* will be replaced by the particle's new position. After updating *pbest*, *gbest* of each particle is updated in the same way by comparing *gbest* with the *pbest* of the particle and its neighbours.

By adding the second situation, the proposed updating mechanism is expected to avoid the limitation of traditional updating mechanism. Where available, it will always select a better feature subset to be the *pbest* or *gbest*, which either has better classification performance or the same classification performance with a smaller number of features. This can help the algorithm filter out redundant features and make the feature subset with good classification performance and a small number of features to be the *leader* (*pbest* or *gbest*) of each particle and the whole swarm.

Note that in GP, each individual can be represented as a tree. The size of the trees can be considered in the selection process, known as parsimony pressure [11]. The parsimony pressure seems similar to the proposed *pbest* and *gbest* updating mechanism. However, they are different ideas in two aspects. Firstly, the parsimony pressure in GP changes the size of the trees while the proposed *pbest* and *gbest* updating mechanism does not change the size of the particles that is always the total number of features in the dataset. Secondly, the parsimony pressure is to control the size of the trees in GP, which can be used in any problem domain, but the number of features considered in the proposed *pbest* and *gbest* updating mechanism is particularly for feature selection problems to optimise one of the two main objectives, i.e., minimising the number of features.

Algorithm 1. The pseudo-code of the proposed algorithm (IniPG)

```
begin
    initialise most of the particle using small feature subsets and the others
    particles using relatively large feature subsets;
    initialise the velocity of each particle;
    while Maximum Iterations or the stopping criterion is not met do
        evaluate the fitness of each particle on the Training set;
        for i=1 to Population Size do
            if fitness of particle i (x_i) is better than that of pbest then
                pbest = x_i ;                    // Update the pbest of particle i
            else if fitness of x_i is the same as pbest and |x_i| < |pbest| then
                pbest = x_i ;                    // Update the pbest of particle i
            if fitness of pbest of any neighbour is better than that of gbest then
                gbest = pbest ;                  // Update the gbest of particle i
            else if fitness of pbest of any neighbour is the same as gbest and
            |pbest| < |gbest| then
                gbest = pbest ;                  // Update the gbest of particle i
        for i=1 to Population Size do
            update the velocity and the position of particle i
    calculate the classification accuracy of the selected features on the Test set;
    return the position of gbest (the selected feature subset);
    return the training and test classification accuracies;
```

Based on the new initialisation strategy and updating mechanism, a new feature selection algorithm is proposed named IniPG. The pseudo-code of IniPG can be seen in Algorithm 1. PSO has two versions, which are continuous PSO [14] and binary PSO [9], but binary PSO has potential limitations [10]. Therefore, we will use continuous PSO to propose a novel feature selection algorithm. The representation of a particle is a "n" bits string, where "n" is the total number of features. The position value in each dimension (x_{id}) is in [0,1]. A threshold θ is needed to compare with the value of x_{id}. If $x_{id} > \theta$, the dth feature is selected. Otherwise, the dth feature is not selected.

3 Design of Experiments

3.1 Benchmark Techniques

To examine the performance of the proposed algorithm (IniPG), two traditional wrapper feature selection methods and two PSO based algorithms (ErRt and 2Stage) as benchmark techniques in the experiments.

The two traditional methods are linear forward selection (LFS) [5] and greedy stepwise backward selection (GSBS), which were derived from SFS and SBS, respectively. More details about LFS can be seen in the literature [5] and GSBS starts with all available features and stops when the deletion of any remaining

Table 1. Datasets

Dataset	#Features	#Classes	#Instances	Dataset	#Features	#Classes	#Instances
Wine	13	3	178	Zoo	17	7	101
Wisconsin Breast Cancer (Diagnostic) (WBCD)	30	2	569	Vehicle	18	4	846
Ionosphere	34	2	351	Lung	56	3	32
Hillvalley	100	2	606	Madelon	500	2	4400

feature results in a decrease in classification performance. ErRt only uses the classification error rate as the fitness function. 2Stage [18] employs a two-stage fitness function to optimise the classification in the first stage and take the number of features into account in the second stage [18]. Binary PSO was used in [18], but continuous PSO is employed in this paper to keep consistent with ErRt and IniPG for fair comparisons.

3.2 Datasets and Parameter Settings

Eight datasets (Table 1) are chosen from the UCI machine learning repository [4], which have different numbers of features, classes and instances. For each dataset, the instances are randomly divided into two sets: 70% as the training set and 30% as the test set.

K-nearest neighbour (KNN) was used in the experiment and K=5 (5NN). Weka [7] is used to run the experiments of using LFS and GSBS. All the settings in LFS and GSBS are kept to the defaults except that backward search is chosen in GSBS. The parameters of PSO in ErRt, 2Stage and IniPG are set as follows: $w = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{max} = 6.0$, population size is 30, and the maximum iteration is 100. The fully connected topology is used. These values are chosen based on the common settings in [14]. According to our previous experiments, the threshold θ is set as 0.6 in the three PSO based algorithms. In IniPG, a major part of the swarm (2/3) is initialised using around 10% of the total number of features. The other minor part of the swarm (1/3) is initialised using more than half of the total number of features, where a random number (e.g. m, where m is between half and the total number of features) is firstly generated and m features is randomly selected to initialise this particle.

For each dataset, each experimental test has been conducted for 40 independent runs. A statistical significance test, T-test, is performed between their classification performances and the significance level was selected as 0.05.

4 Experimantal Results and Discussions

Table 2 shows the experimental results of the proposed algorithm and the benchmark techniques. "All" means that all features are used for classification. "NO." represents the average number of features selected. "Ave", "Best" and "StdDev" indicate the average, the best and the standard deviation of the 40 test accuracies in ErRt, 2Stage or IniPG. "T-test" shows the result of the T-test, where "+" ("-") means that the classification performance of a benchmark technique is significantly better (worse) than that of IniPG. "=" indicates they are similar.

Table 2. Experimantal Results

Dataset	Method	NO.	Ave(Best)	StdDev	T-test	Dataset	Method	NO.	Ave(Best)	StdDev	T-test
Wine	All	13	76.54		-	Zoo	All	17	80.95		-
	LFS	7	74.07		-		LFS	7	74.07		-
	GSBS	8	85.19		-		GSBS	8	85.19		-
	ErRt	8	95.96 (100)	1.83E-2	=		ErRt	9.18	95.5 (97.14)	90.3E-4	=
	2Stage	8	95.96 (100)	1.83E-2	=		2Stage	9.18	95.5 (97.14)	90.3E-4	=
	IniPG	6.78	95.12 (98.77)	1.87E-2			IniPG	6.58	95.52 (97.14)	71.3E-4	
WBCD	All	30	92.98		-	Vehicle	All	18	83.86		-
	LFS	10	88.89		-		LFS	9	83.07		-
	GSBS	25	83.63		-		GSBS	16	75.79		-
	ErRt	13.42	93.39 (94.74)	55.8E-4	-		ErRt	9.52	85 (87.01)	79E-4	=
	2Stage	5	93.54 (94.74)	75.1E-4	-		2Stage	8.65	84.95 (87.01)	77.9E-4	=
	IniPG	3.45	94.09 (94.74)	82.5E-4			IniPG	10.28	85.31 (87.01)	95.5E-4	
Ionosphere	All	34	83.81		-	Lung	All	56	70		-
	LFS	4	86.67		-		LFS	6	90		+
	GSBS	30	78.1		-		GSBS	33	90		+
	ErRt	12.58	88.4 (93.33)	2.14E-2	+		ErRt	27.35	72 (80)	6E-2	-
	2Stage	12.05	88.14 (91.43)	1.89E-2	+		2Stage	27.38	72.25 (90)	6.89E-2	-
	IniPG	3.2	87.14 (91.43)	1.88E-2			IniPG	6.22	78.75 (90)	6.4E-2	
Hillvalley	All	100	56.59		-	Madelon	All	500	70.9		-
	LFS	8	57.69		=		LFS	7	64.62		-
	GSBS	90	49.45		-		GSBS	489	51.28		-
	ErRt	47.32	57.54 (61.81)	1.52E-2	=		ErRt	258.1	76.55 (79.49)	1.22E-2	-
	2Stage	47.05	57.57 (61.81)	1.55E-2	=		2Stage	256.48	76.52 (79.36)	1.26E-2	-
	IniPG	12.72	57.95 (60.71)	1.48E-2			Initia	216.4	78.49 (84.23)	3.23E-2	

4.1 Results of Benckmark Techniques

Results of LFS and GSBS: according to Table 2, LFS selected a smaller number of features and achieved a similar or higher classification accuracy than using all features in most cases. GSBS could reduce the number of features, but only achieved better classification performance on a few datasets. In most cases, LFS outperformed GSBS in terms of both the number of features and the classification performance. The results indicate that LFS as a forward selection algorithm is more likely to obtain good feature subsets with a small number of features than GSBS (backward selection) because of different starting points. Feature subsets selected by GSBS may still have redundancy. This also suggests that utilising the advantages of both forward selection and backward selection can improve the performance of a feature selection algorithm, which motivates the proposal of the new initialisation strategy in this work.

Results of ErFs: according to Table 2, in almost all datasets, ErRt achieved similar or better classification performance than using all features, and the evolved feature subsets only included around half of the available features. This suggests that PSO as an evolutionary search technique can be successfully used for feature selection problems.

Results of 2Stage: according to Table 2, 2Stage evolved feature subsets with around half (or less) of the available features and achieved better classification performance than using all features in almost all cases. 2Stage outperformed ErRt in almost all cases. However, 2Stage attempted to find a trade-off between

the classification performance and the number of features, which means the reduction of the number of features might decrease the classification performance.

4.2 Results of IniPG

According to Table 2, in all datasets, IniPG evolved feature subsets that selected less than half (or even close to than 10% in four datasets) of the available features, but achieved significantly better classification performance than using all features.

Comparisons Between IniPG and Two Traditional Methods (LFS and GSBS): in almost all datasets, IniPG achieved significantly better or similar classification performance to LFS, although the number of features is slightly larger in some cases. Comparing IniPG with GSBS, the number of features in IniPG is smaller than GSBS in all datasets and the classification performance of IniPG is significantly better than GSBS in 7 of the 8 datasets. This suggest that IniPG as a PSO based algorithm can search the solution space more effectively than both LFS and GSB. The initialisation strategy movitated by both forward selection and backward selection can help IniPG take the advantages of both forward selection and backward selection to obtain feature subsets with a smaller number of features and better classification performance than both LFS and GSB.

Comparisons Between IniPG and ErRt: according to Table 2, IniPG selected feature subsets including smaller numbers of features and achieved significantly better or similar classification performance than ErRt in almost all datasets (except for the Ionosphere dataset, where the number of features in IniPG is around one fourth of that in ErRt). This suggests that although ErRt and IniPG shared the same fitness function (Equation 3), the proposed initialisation strategy and *pbest* and *gbest* updating mechanism can help IniPG to effectively eliminate the redundant and irrelevant features to obtain a smaller feature subset with significantly better classification performance than ErRt.

Comparisons Between IniPG and 2Stage: according to Table 2, in almost all datasets, the classification performance of IniPG is significantly better or similar to that of 2Stage and the number of features is smaller. The reason might be that the fitness function in the second stage in 2Stage aims to find a balance between the classification performance and the number of features. Therefore, the reduction of the number of features will also decrease the classification performance. In IniPG, the fitness function only includes the classification performance during the whole evolutionary process. This ensures that the reduction of the number of features in IniPG will not reduce the classification performance. Meanwhile, the proposed initialisation strategy and *pbest* and *gbest* updating mechanism can help IniPG further remove the irrelevant or redundant features to reduce the number of features, which in turn could increase the classification

performance. In addition, compared with 2Stage, another advantage of IniPG is that it does not need a predefined parameter to balance the relative importance of the classification performance and the number of features.

Note that simply increasing the number of iterations cannot help ErRt and 2Stage achieve the same performance obtained by IniPG. The main reason is that ErRt does not consider the number of features in the fitness function and 2Stage takes a trade-off between the classification performance and the number of features. IniPG simulates both forward and backward selection to duplicate their advantages, which helps IniPG pay more attention to small feature subsets, but does not miss the large feature subsets with high classification performance. Meanwhile, because of the new updating mechanism, for two feature subsets with the same classification performance, IniPG will select the smaller one as the new *pbest* or *gbest*. ErRt and 2Stage using traditional updating mechanism will not do this during the evolutionary training process. Therefore, ErRt and 2Stage can not achieve as good performance as IniPG in almost all situations.

4.3 Analysis on Computational Time

All the five methods used in the experiments are wrapper based feature selection approaches. Therefore, most of their computational time is spent on the fitness evaluation, which regards the training and testing classification processes.

LFS usually used less time than the other four methods because the forward selection strategy starts with a small number of features and the evaluation of a small feature subset takes less time than a large feature subset. GSBS cost less time than other three PSO based algorithms (ErRt, 2Stage and IniPG) on the datasets with a small number of features, but more time on the datasets with a large number of features, such as the Madelon datasets. The reason is that GSBS starts with the full set of features, which needs much longer time for each evaluation. The number of evaluations in GSBS substantially increases in such large datasets while the number of evaluations in PSO based algorithms is still the same. Generally, 2Stage cost less time than ErRt because the size of the feature subsets evolved by 2Stage is smaller than ErRt during the evolutionary training process. For the same reason, the computational time of IniPG is less than both of ErRt and 2Stage.

5 Conclusions

This paper proposes a new PSO algorithm for feature selection problems (IniPG). In IniPG, *a new initialisation strategy* was proposed based on the ideas of two traditional feature selection methods (forward selection and backward selection) to utilise the advantages of these two methods. Meanwhile, *a new pbest* and *gbest updating mechanism* was proposed to overcome the limitation of the traditional updating mechanism in order to ensure the feature subset with the highest classification performance and the smallest number of features become the new *pbest* or *gbest*. IniPG was examined and compared with two traditional feature

selection algorithms (LFS and GSBS), a PSO based algorithm with only the classification error rate as the fitness function (ErRt) and a PSO based two-stage algorithm (2Stage). Experimental results show that in almost all datasets, IniPG achieved significantly better classification performance than LFS and GSBS, although the number of features is larger than LFS in some cases. In almost all cases, IniPG outperformed ErRt and 2Stage in terms of the number of features and the classification performance, and used less computational time.

In the future, we will further tune the PSO algorithm for feature selection problems. We intend to work on multi-objective PSO for feature selection in classification problems. We will also investigate whether using a given learning algorithm in a wrapper feature selection approach can select a good or near-optimal feature subset for other learning algorithms.

References

1. Banerjee, M., Mitra, S., Banka, H.: Evolutionary rough feature selection in gene expression data. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(4), 622–632 (2007)
2. Chuang, L.Y., Chang, H.W., Tu, C.J., Yang, C.H.: Improved binary PSO for feature selection using gene expression data. Computational Biology and Chemistry 32(29), 29–38 (2008)
3. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis 1(4), 131–156 (1997)
4. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
5. Gutlein, M., Frank, E., Hall, M., Karwath, A.: Large-scale attribute selection using wrappers. In: IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2009), pp. 332–339 (2009)
6. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. The Journal of Machine Learning Research 3, 1157–1182 (2003)
7. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. The Mathematical Intelligencer 27, 83–85 (2005)
8. Huang, C.L., Dun, J.F.: A distributed PSO-SVM hybrid system with feature selection and parameter optimization. Application on Soft Computing 8, 1381–1391 (2008)
9. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108 (1997)
10. Khanesar, M., Teshnehlab, M., Shoorehdeli, M.: A novel binary particle swarm optimization. In: Mediterranean Conference on Control Automation, pp. 1–6 (2007)
11. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 829–836 (2002)
12. Marill, T., Green, D.: On the effectiveness of receptors in recognition systems. IEEE Transactions on Information Theory 9(1), 11–17 (1963)
13. Neshatian, K., Zhang, M.: Using genetic programming for context-sensitive feature scoring in classification problems. Connection Science 23(3), 183–207 (2011)
14. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation (CEC 1998), pp. 69–73 (1998)

15. Unler, A., Murat, A.: A discrete particle swarm optimization method for feature selection in binary classification problems. European Journal of Operational Research 206(3), 528–539 (2010)
16. Wang, H., Li, H., Liu, Y., Li, C., Zeng, S.: Opposition-based particle swarm algorithm with cauchy mutation. In: IEEE Congress on Evolutionary Computation (CEC 2007), pp. 4750–4756 (2007)
17. Whitney, A.: A direct method of nonparametric measurement selection. IEEE Transactions on Computers C-20(9), 1100–1103 (1971)
18. Xue, B., Zhang, M., Browne, W.N.: New fitness functions in binary particle swarm optimisation for feature selection. In: IEEE Congress on Evolutionary Computation (CEC 2012), pp. 2145–2152 (2012)

CodeMonkey; a GUI Driven Platform for Swift Synthesis of Evolutionary Algorithms in Java

Reza Etemadi, Nawwaf Kharma, and Peter Grogono

Electrical and Computer Engineering Department, Concordia University, Montreal (QC), Canada H3G 1M8
{r_etemad@encs,kharma@ece,grogono@cse}.concordia.ca

Abstract. CodeMonkey is a GUI driven software development platform that allows non-experts and experts alike to turn an evolutionary algorithm design into a working Java program, with a minimal amount of manual code entry. This paper describes the concepts behind CodeMonkey, its internal architecture and manner of use. It concludes with a simple application that exhibits its utilization for multi-dimensional function optimization. CodeMonkey is provided free of charge, for non-commercial users, as a plug-in for the Eclipse platform.

Keywords: Evolutionary Algorithm, Java Language, Eclipse Platform, GUI Application.

1 Introduction

CodeMonkey is a GUI-driven software platform that allows non-expert users to generate an executable Java implementation of a custom-designed Evolutionary Algorithm, meant for a particular optimization or design application.

1.1 Review

There are many types of Evolutionary Algorithms (EA) including Genetic Algorithms (GA), Genetic Programming (GP), Evolutionary Strategies (ES) and Evolutionary Programming (EP). Software platforms differ in: scope (some are more generic while others are specialized to certain domains); performance and scalability (based on the language or platform they employ); usability (whether it is just a library, framework, application or platform). A few widely-used EA development aides follow.

GEATbx [1] (Genetic and Evolutionary Algorithm Toolbox) is a Matlab tool, and one of the few packages that cover the four main flavors of EA. It allows users to define homogeneous genotypical representations (i.e., lists of one type of variable); it has limited support for heterogeneous genotypes (with different types in the same structure) [2]. It offers a GUI for novice users. It has a limited selection of genetic variation operations. It is a proprietary package and is not extendable by third parties.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 439–449, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Evolving Objects (EO) [3] is a template-based C++ open-source framework for writing stochastic optimization programs. It supports homogeneous genotypical representations (but not integers). It does not offer readily usable heterogeneous genotypes. Many selection mechanisms are provided. It does not allow GUI based customization. It is open-source and free to use under GPLv2. It is extendable by third-parties.

DREAM (Distributed Resource Evolutionary Algorithms Machine) [4][5] software framework (called Java Evolving Object) defines many genotypes together with matching variation operations. It does not provide heterogeneous genotypes. It has a GUI for non-programmers for both I/O interaction and problem definition. It is open-source and is offered under a GPL license. The platform has a distributed architecture, and the framework has an API that facilitates distributed implementations.

Watchmaker Framework for Evolutionary Computation [6] is a framework for implementing evolutionary algorithms in Java. It is used in the Apache Mahout Project, and some specialized frameworks, such as GEP4J. It does not provide predefined genotypes. There is a GUI for monitoring progress, but it does not offer a GUI for non-experts to generate code. There are a limited number of variation operators. It is open-source and available under Apache software license, and it is easily extendable.

Table 1. Camparative Summary of Common EA Platforms

Name	GEATbx	Evolving Objects	DREAM	Watchmaker	JGAP
Programming Language	Matlab	C++	EASEA /Java	Java	Java
Type	Library	Framework	Platform	Framework	Framework
Homogeneous Genotypes	Real, Integer, Binary, Permutation	Real, Binary, Permutation	Real, Integer, Binary, Tree	None	Real, Integer, Binary, String & more
Heterogeneous Genotypes	None	None	None	None	None
Selection Types	4	9 (more pluggable)	11 (more pluggable)	6 (more pluggable)	4 (more pluggable)
GUI for Novice users	Yes	No	Yes	No	No
Open Source	No	Yes	Yes	Yes	Yes
Extendable	No	Yes	Limited	Yes	Yes
Licencing /Pricing	Multi-tier /Not-free [8]	GPLv2 /Free	GPL /Free	Apache v2.0 /Free	LGPL;Mozilla /Free

JGAP (Java Genetic Algorithms Package) [7] is a GA and GP component provided as a Java framework. It provides basic genetic mechanisms that can be used to apply evolutionary principles to problem solutions. It has many ready-to-use genotypes defined, but does not offer heterogeneous genotypes. It also does

not provide a GUI for end-users to generate code; only expert users, who must learn the framework and have prior knowledge of GA or GP, can make use of JGAP. It is, however, an open-source resource, free and extendable under LGPL and Mozilla licenses. Table 1 provides a brief comparison of the EA platforms outlined in this section.

The comparison table demonstrates that only GEATbx and DREAM offer a customization GUI for non-expert users. None of them attempts to provide support for heterogeneous genotypes. In contrast, we provide an easy-to-learn and use GUI-driven platform for the generation of EAs of different flavors and for many target application. It supports both homo- and heterogeneous genotypes with appropriately defined variation operators. It also offers a great degree of flexibility in both offspring generation and survivor selection; a full description of its internal design and manners of use follows. The paper concludes with a simple use example.

2 Design and Implementation

CodeMonkey (CM) is a project that takes advantage of modern features of the Java language, such as Generics and Annotation, and combines that with the power and ease of use of the Eclipse platform, to provide both a framework for expert users and an Eclipse plug-in application for non-expert developers of Evolutionary Algorithms.

The framework aspect of CM is not unique, but Eclipse integration and its step by step GUI wizard allow users with little knowledge of programming to generate serious EA programs that are readily executable.

2.1 Concept

This section offers a description of CMs main configuration steps. This is interleaved with explanations of the main conceptual innovations that were employed in order to make CM as generic as it is.

Genotype Representation: CM divides genotype representations into two categories: homogeneous and heterogeneous. A homogeneous genotype is a collection of genes with identical types (e.g., Boolean, integer or real) while a heterogeneous genotype is a collection of homogeneous genes of different types. CM supports basic homogeneous genotypes and allows easy definition of heterogeneous genotypes.

Termination Criteria: CM places the termination conditions which can be combined under three categories: (a) the Goal Achieved category, which means that an acceptable level of fitness has been attained by at least one individual in the population; (b) the Stagnation Reached category, which means that the improvement in fitness over a preset number of generations is too low to justify continuation; (c) the Resources Exhausted category, which means that a preset limit on a computational resource, such as processing time, has been reached or breached.

Parent Selection: there is a wide spectrum of algorithms for parent selection [9], from fully deterministic (such as Truncation) to fully probabilistic (such as

Random), which are typically applied to two individuals. Prior to the actual application of any selection method, the raw fitness of an individual can be transformed (e.g., via ranking) into a different value: it is this new value that is used by the selection method. Regarding selection methods, a unique contribution of CM is the way it unifies many different parent selection mechanisms into one window-based selection generic algorithm. This algorithm employs a selection window, which can be as large as the population size or as small as two individuals. Within a window of a certain size, the picking of an individual from the population can occur on a deterministic or probabilistic basis. The result is a parent, which is deposited into the parent pool. For example, binary tournament selection can be viewed as deterministic selection of the fittest individual from a window of size two. While proportional selection can be seen as probabilistic selection from a window that includes the whole population.

Variation Operations: there are crossover and mutation operations suited to different genetic representations. In CM, different operations are defined for different homogeneous and heterogeneous genotypes. Another contribution of CM is that any number of variation operations can be used with different probabilities and in different sequences along one or more paths linking the parent pool to the offspring pool. This allows users of CM to define a GA-like single sequence of variation operations of say crossover followed by mutation, or alternatively define a GP-style tree of variation operations, with crossover working in parallel with mutation to generate offspring.

Survivor Selection: it allows the generation of the next population using individuals from the current population and/or offspring pool. In CM, all selection mechanisms used for parent selection are available to survivor selection. The difference is that we can apply the selection window to any one or both of the current population and the offspring pool. In addition, CM allows the use of elitism and injection.

2.2 The Framework

To explore CodeMonkeys framework, we start by describing its architecture and data model. The class diagram in figure 1 presents the core package of the framework, which includes the main classes described below.

Class Phenotype represents an individual solution. In the core package this is an abstract class that has two other elements, Genotype and Fitness (mapped through generics). In any implementation, a subclass of this class will be needed for representing individuals.

Class Genotype is the class that represents the genetic encoding of Phenotype. It is a Java interface that extends Java Collection interface. For Boolean, integer and real types, the framework provides implementation. Several variation operators are available in the framework for each genotype implementation. These classes are also used for building homogeneous and heterogeneous genotypes in the CM application.

Class Fitness represents the suitability of the phenotype as a solution in comparison to other phenotypes. This is also defined as a Java interface in the

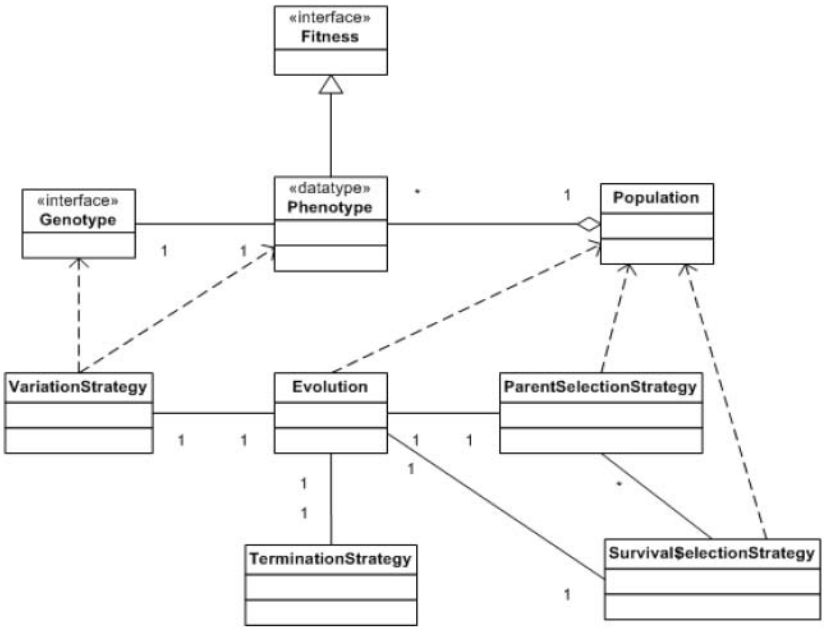


Fig. 1. CodeMonkeys Class Diagram

framework that extends Java Comparable interface. This interface uses Java generics to accept any subclasses of the Java Number class that implements the Comparable interface.

Class Population represents a collection of Phenotypes. This class (or its subtypes) can be registered to represent the initial population, the parent pool, the offspring pool as well as the next generation. This class has many utility methods for random selection and sorting as well as calculating various statistics of the population (based on fitness and other attributes).

Class TerminationStrategy is an abstract class that represents the termination criteria in the framework. All possible criteria that are used in the CM application are provided in this class. Any concrete subclass can be registered in an implementation. It is invoked periodically to see if the evolutionary process should terminate.

Class ParentSelectionStrategy is the class that realizes the unitary parent selection approach described above. There are three subclasses of this class: one for Truncation selection, one for Proportional selection and a third for Random selection. Any concrete subclass of ParentSelectionStrategy that is registered will be invoked to create the parent pool.

Class VariationStrategy is the abstract class of all variation operations in the framework. Any concrete subclass will invoke the variation method that is implemented in the Genotype with the desired probability and sequence. The class must be registered before it can be invoked to create the offspring pool from the parent pool.

Class SurvivalSelectionStrategy is the abstract class for the survivor selection process in the framework. It internally relies on the ParentSelectionStrategy. A concrete subclass will need to define what percentage of the next population comes from what available population and based on which selection mechanism. The subclass must be registered before it is invoked to create the next generation from the current population and/or the offspring pool.

Finally, class Evolution is the orchestrator of the evolutionary process: the general logic of evolution is implemented here. Any implementation based on the framework will create a concrete subclass of this class and include a main method, so it can be called as a Java application. All above-mentioned registrations of data types, strategies and pool sizes need to be defined in the concrete subclass. Once all necessary elements are registered, the class can be executed to launch the evolutionary process.

2.3 Plug-in Application

The CodeMonkey application is built on top of the Eclipse platform. It uses Eclipses plug-in architecture [10] to create a GUI-based application. The user of CM employs a GUI to provide customizing inputs reflecting a specific EA flavor and target application. Hence, the CM application uses the Eclipse JDT (Java Development Tools) API to create the necessary code, which in turn completes the CM framework. The user can then launch the generated Java program in Eclipse.

Two types of users can use CM to customize and generate an EA in Java: novice and expert. Expert users can directly work with the framework by using existing functionalities or extending them and adding new implementations. Novice users are asked to provide the CM with customizing inputs, which allows CM to generate a Java program that implements a specific EA flavor for a specific EA application. The only part of CM that necessitates the provision of either (1) actual code or (2) a link to an external program or function is the fitness function.

As shown, once the CM plug-in application is launched, the first step is defining the genotype, followed by configuring initialization. Hence, the user defines how fitness will be calculated. This is the only step that necessitates manual code entry or external communications. The next step is defining the termination criteria. This is followed by customizing parent selection. The remaining two steps are defining how variation operations are applied and how the next generation is created. Once those steps are completed, the generated code is compiled and can be run.

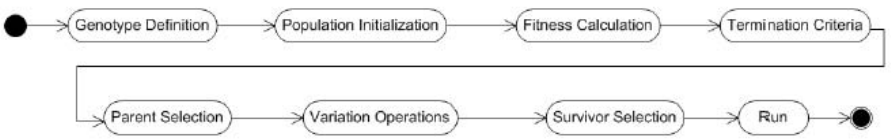


Fig. 2. Activity Diagram of CM Plug-in Application

2.4 Program Execution

Whether the code is generated by the CM application or directly entered into the generated program, the execution of the resulting program follows the process exhibited in figure 3.

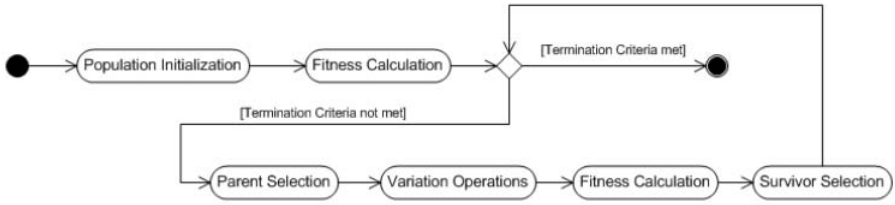


Fig. 3. Activity Diagram of Code Execution

First comes initialization of the first generation, followed by fitness calculation. Hence, the termination criteria are evaluated. As long as the termination criteria are not satisfied the process goes through parent selection, application of variation operations (to generate offspring), evaluating the fitness of the offspring and hence, generating the next population.

3 Example

In this section, we demonstrate how an end-user can use CM to implement an EA solution to a specific multi-dimensional optimization problem.

The Ackley problem [11] is a n-dimensional minimization problem. The goal is to find $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ within $x_i \in \{-32.768, 32.768\}$ that minimizes the function:

$$F(x) = -20.\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}.\sum_{i=1}^{n}\cos(2\pi.x_i)\right) + 20 + e \quad (1)$$

3.1 Solution

To outline the solution we start by defining the genotype. In this case it will be a list of real-valued numbers, one per dimension. The dimensions can be initialized randomly to values from a limited range. The fitness function is the formula itself. The termination criteria are a combination of goal achieved, evolutionary stagnation and resource exhaustion. For parent selection and survivor selection many types of deterministic and probabilistic selection methods can be selected. To generate offspring, a number of crossover and mutation operators can be used. We configured CM three different ways, with details of the first (reference) configuration presented in Table 2; the difference between the other configurations and the reference is described as well.

3.2 Implementation

The following table presents a comprehensive summary of the parameters and their selected settings for CMs reference configuration C1.

Table 2. List of Parameters and Their Values for the First Configuration (C1) of CM

Genotype Definition	Length	10
	Type	Real
	Lower Bound Value	-32.0 (the same for all genes)
	Upper Bound Value	32.0 (the same for all genes)
	Selected Variation Operators	(Discrete Recombination, Continuous Recombination, Convex Recombination, Local Crossover, One-Position Mutation, Creep Mutation)
Population Initialization	Population Size	300
	Random Generator	Uniform
Fitness Calculation	Mechanism	Internal (formula entered manually)
	Type	Minimization
	Target Fitness	0.0
Termination Criteria	Goal Achieved	Stops if Target Fitness is reached
	Stagnation Reached	Stops if no progress over 1000 generation
	Resource Exhausted	Stops if generation reached 3000
Parent Selection	Window Input Size	20
	Window Output Size	15
	Type of Selection	Proportional (w. replacement)
	Fitness Transformation	Ranking
	Parent Pool Size	150
Variation Operations	Offspring Pool Size	150
Survivor Selection	Selection Type	Proportional (w.o. replacement)
	Elitism (%)	5%
	Re-initialization (%)	5%

C2 alters population size to 100 and offspring pool size to 50; C3 makes parent selection deterministic instead of the original probabilistic mode in C1.

3.3 Results

The best result (owing to configurations C1 and C2) was a fitness value of 8.88×10^{-16}, which is practically zero. The genotype of the best individual is $\{4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}, 4.9 \times 10^{-324}\}$, which is vector 0. This individual was found at generation 511 (in case of C1) and at generation 4421 (in case of C2). Configuration C3 failed to return an optimal solution even after 7000 generations (which was set as a stop condition). A time course for the evolution of best fitness for the three configurations is presented in figure 4.

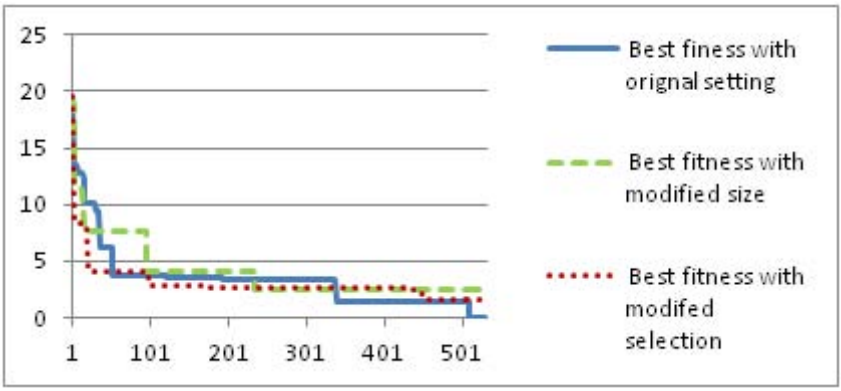


Fig. 4. The History of Evolution of Best Fitness Over 530 Generations

The flexibility of the framework allows the user to go back to any of the steps and change the settings for that particular part of the evolutionary process. A re-run of the code will incorporate the changes and affect the results. The following table shows the results of the three different configurations: C1, C2 and C3.

Table 3. Results for Different Configurations

	C1	C2	C3
Best Fitness @ Gen. 500	1.3860	2.4721	1.6011
Best Achieved Fitness (& When)	8.88×10^{-16} (@ Gen. 511)	8.88×10^{-16} (@ Gen. 4421)	2.79×10^{-1} (@ Gen.7000)

The first row presents the best fitness achieved by a particular configuration at generation 500. The second row contains the best fitness achieved after 7000 generations or less- that is if the optimal solution is found early.

The second column has the results of reference configuration C1. The results of C2 as shown in the third column of Table 3 demonstrate that a reduction of both the population size and offspring pool by 2/3 lead to an 8 fold increase, compared to C1, of the time necessary to reach the optimal solution. The final configuration (C3) has the same population size and offspring pool size as C1, but it employs deterministic instead of C1s probabilistic parent selection. As a result, C3 achieves a higher best fitness than C2 does initially, but eventually returns a worse best fitness value than both C1 and C2.

In all cases, the manner in which evolution progressed over time was typical and the way in which the results differed was explainable (e.g., a strictly deterministic selection method performing badly on highly multi-modal fitness landscapes).

4 Conclusion

CodeMonkey provides a flexible and easy way to customize and complete (with a fitness function) a generic evolutionary algorithm, to generate a customized Evolutionary Algorithm that reflects the users target application as well as his preferred EA style and configuration. The combination of feature-rich Java and Eclipses popularity make CodeMonkey a handy tool for both expert and non-expert developers of EA applications.

References

1. Genetic and Evolutionary Algorithm Toolbox for Matlab, http://www.geatbx.com
2. Geatbx Parameter Optimization, http://www.geatbx.com/docu/algindex-09.html#P1058_123869
3. Evolving Objects (EO), http://eodev.sourceforge.net
4. Back, T., Schoenauer, M., Sebag, M., Eiben, A., Merelo, J., Fogarty, T.: A Distributed Resource Evolutionary Algorithm Machine (DREAM). IEEE Transaction on Evolutionary Computation 2, 951–958 (2000)
5. DREAM, http://www.soc.napier.ac.uk/~benp/dream/dream.htm
6. Watchmaker Framework, http://watchmaker.uncommons.org
7. Java Genetic Algorithm Package, http://jgap.sourceforge.net
8. Geatbx Pricing, http://www.geatbx.com/prices.html
9. Dumitrescu, D., Lazzerini, B., Jain, L., Dumitrescu, A.: Evolutionary Computation, ch. 3–5 (2000)
10. Notes on the Eclipse Plug-in Architecture, http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html
11. Bäck, T.: Ackley's Function, in Evolutionary algorithms in theory and practice, pp. 142–143. Oxford University Press (1996)

Multi-Objective Optimizations of Structural Parameter Determination for Serpentine Channel Heat Sink

Xuekang Li[1], Xiaohong Hao[1], Yi Chen[1,2], Muhao Zhang[1], and Bei Peng[1,*]

[1] School of Mechanical, Electronic and Industrial Engineering, University of Electronic Science & Technology of China, 2006 Xiyuan Ave, Chengdu, SC 611731, P.R. China.
[2] School of Engineering & Built Environment, Glasgow Caledonian University, Glasgow G4 0BA Scotland, UK.
beipeng@uestc.edu.cn

Abstract. This paper presents an approach for modeling and optimization of the channel geometry of a serpentine channel heat sink using multi-objective genetic algorithm. A simple thermal resistance network model was developed to investigate the overall thermal performance of the serpentine channel heat sink. Based on a number of simulations, bend loss coefficient correlation for 1000<Re<2200 was obtained which was function of the aspect ratio (a), ratio of fins width to channel width (b). In this study, two objectives minimization of overall thermal resistance and pressure drop are carried out using multi-objective genetic algorithms. The channel width, fin width, channel height and inlet velocity are variables to be optimized subject to constraints of fixed length and width of heat sink. The study indicates that reduction in both thermal resistance and pressure drop can be achieved by optimizing the channel configuration and the inlet velocity.

Keywords: heat sink, serpentine channel, bend loss coefficient, multi-objective optimizations.

1 Introduction

Heat sinks are used to remove heat from various devices and transfer it to the ambient effectively. The geometry design of heat sink is to dissipate as much heat as possible under limitations such as pumping power and weight of its own. With the growing demand for high heat flux dissipation, liquid-cooled heat sink is used increasingly widely. Therefore a large number of researches have been done to optimize parameters of heat sink under various design conditions.

In the earlier study, Tuckerman and Pease [1] designed a micro-channel heat sink. After this landmark paper, lots of optimization works have been carried out concerning micro-channel heat sink. Knight [2] presented a dimensionless form of governing equations for fluid dynamic and heat transfer for both laminar and turbulent flow and used it to determine the geometry of micro-channel heat sink. The optimized results

* Corresponding author.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 449–458, 2013.
© Springer-Verlag Berlin Heidelberg 2013

improved the thermal resistance of heat sinks reported by previous investigators ranging from 10 to 35% in values. Perret [3] and Biswal [4] used analytic model to optimize the single-phase liquid cooled micro-channel heat sink and realized the cooling devices.

In recent years, genetic algorithms (GAs) are frequently used in the design of thermal systems [5-10]. In this paper, serpentine channel heat sink is optimized to reduce the overall thermal resistance and the pressure drop with the constraint of its physical parameters. NSGA-II [11] is applied to obtain the Pareto-optimal solutions. The variables are channel width, fin width, channel height and inlet velocity. The thermal resistance model and pressure drop model of serpentine heat sink are thus established. Lastly numerical method is used to confirm the solutions.

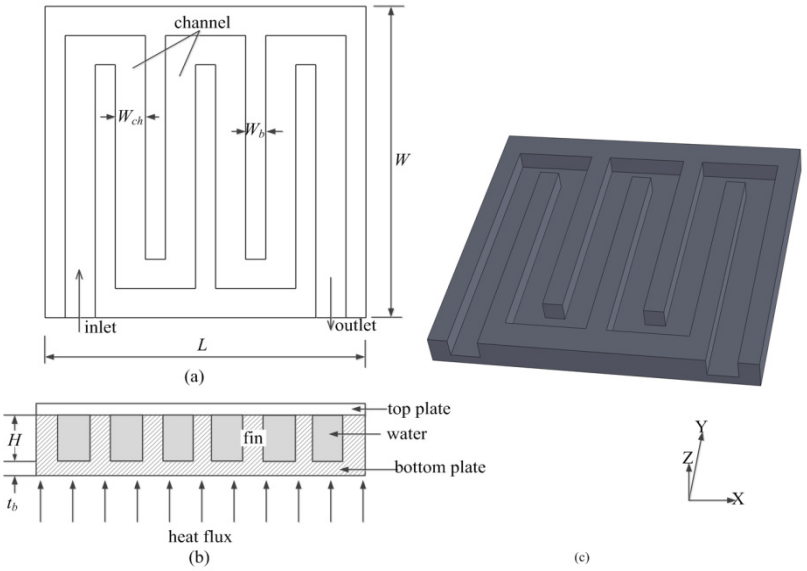


Fig. 1. (a) Flow field of the serpentine channel heat sink; (b) Cross-section of heat sink separated into 10 units; (c) Three dimensional domain of the heat sink

2 The Methodology

The structure of a serpentine channel heat sink is shown in Fig. 1. It consists of a uniform heat flux from the base plate. The structure has n channels and (n-1) fins. The top plate is used solely for containing the coolant flow which is assumed to be insulated. The heat is taken away by the coolant. The two factors being considered to present performance of the heat sink are the total thermal resistance and pressure drop of the serpentine channel. It is imperative to determine optimum geometrical parameters of the heat sink that would result in desired thermal resistance and pressure drop for cost effective and efficient thermal solutions.

2.1 Thermal Resistance

The serpentine channel heat sink is different from parallel channel heat sink for there are several sharp bends. Ignoring the effect of heat transfer coefficient at bends, the total thermal resistance of the heat sink is given by

$$R_{\text{tot}} = \frac{T_{\max} - T_{\text{in}}}{Q} = R_{\text{con}} + R_{\text{cov}} + R_{\text{fluid}} \tag{1}$$

Equation (1) does not take into consideration the spreading thermal resistance caused by different temperature between outlet and inlet or the interface thermal resistance between the device and the heat sink. The conductive thermal resistance is defined by

$$R_{con} = \frac{t}{k_{hs} A_{hs}} \tag{2}$$

The total convection thermal resistance is formulated as

$$R_{\text{cov}} = \frac{1}{hA_s} \tag{3}$$

While the surface area available for heat transfer, A_s, is written as:

$$A_s = nw_c(W - 2bw_c) + 2n\eta aw_c(W - 2bw_c - cw_c) \tag{4}$$

where

$$a = \frac{H}{w_c},\ b = \frac{w_f}{w_c},\ c = \frac{l_s}{w_c},\ \eta = \frac{\tanh(mH)}{mH} \tag{5}$$

The term, m, in eq. (5) is defined by its approximate equality [2]

$$m \approx \sqrt{\frac{h}{k_{hs} b w_c}} \tag{6}$$

The heat transfer coefficient, h, is obtained from the average Nusselt number

$$h = \frac{Nu_m k_f}{D_h} \tag{7}$$

Serpentine channel has several bends which interrupt the hydrodynamic boundary layers. So the thermal profiles must also develop from the bends downstream and the entrance downstream. In this paper, the average Nusselt number is used to take this situation into account. The average Nusselt number for laminar flow is calculated by [12]

$$Nu_m = \left\{ \left[2.22 \left(x^* \right)^{-0.33} \right]^3 + Nu \right\}^{1/3} \tag{8}$$

where

$$x^* = x/\mathrm{Re}\,D_h\,\mathrm{Pr}\,,\ \ Nu = 8.31G - 0.02\,,\ \ \mathrm{Re} = \frac{VD_h}{\nu} \tag{9}$$

$$D_h = \frac{2W_{ch}H}{W_{ch}+H}\ \ ,\ \ G = \left[\left(\frac{W_{ch}}{H}\right)^2 + 1\right] \Bigg/ \left(\frac{W_{ch}}{H} + 1\right)^2 \tag{10}$$

The capacitive thermal resistance is defined by

$$R_{fluid} = \frac{1}{\dot{m}c_p} = \frac{1}{\rho_f W_{ch} H V c_p} \tag{11}$$

where $\dot{m}$ is the mass flow rate of the coolant. This thermal resistance presents the temperature rise from inlet to outlet in bulk fluid.

2.2 Pressure Drop

Pressure drop is another factor used to characterize performance of heat sink. In this structure of channels, pressure drop consists of two components, namely pressure drop due to straight channel friction and that due to the bends. Thus, the total pressure drop is given by:

$$\Delta P = \frac{1}{2}\rho V^2 \left(4 f_{app} \frac{L_t}{D_h} + \sum_{i=1}^{n} \xi_i\right) \tag{12}$$

where f_{app} is the friction loss coefficient in straight section, and the second term is the sum of loss coefficients for n U-bends in the serpentine channels. The serial bends interrupt hydrodynamic boundary periodically, the effect of laminar flow development and re-development is considered [13, 14]. Like the average Nusselt number, the average friction factor in straight pipe is given by[12]:

$$f_{app} = \left\{\left[3.2\left(x^+\right)^{-0.57}\right]^2 + \left(f\,\mathrm{Re}\right)^2\right\}^{1/2} \Big/ \mathrm{Re} \tag{13}$$

where

$$x^+ = x/\mathrm{Re}\,D_h \tag{14}$$

and f is the friction factor of a fully developed laminar flow is calculated using correlation[2]

$$f = \frac{4.70 + 19.64G}{\mathrm{Re}} \tag{15}$$

The bend loss coefficient, ξ_i, could be calculated by correlations proposed by Maharudrayya [15]. It is developed as a function of the Reynolds number, aspect ratio, curvature and the width of fins. The three-regime correlations are valid for the range of some parameters, i) $1/6 < a = H/W_{ch} < 1$, ii) $1 < w_f/D_h < 30$. However, the

structure being considered in this paper may be out of range for these parameters. The depth of channel may be larger than width, namely a>1, and width of fin may be smaller than that of channel, namely $b = w_f / W_{ch}$ <1. The effect of these geometric parameters on the pressure loss is investigated below.

3 Calculation the Bend Loss Coefficient

The geometry studied to calculate the bend loss coefficient was sharp 180° bend(s) in a channel of rectangular cross-section. Each bend was provided with a $30D_h$ long upstream and a same length downstream for flow development. The CFD software was used to simulate the flow condition with three-dimensional domain. The fluid properties corresponded to those of air at atmospheric conditions. The computational domain of the geometry was discretized by at least 500,000 cells. The mesh was refined at the bend, since the velocity gradients near the bend was higher and a better resolution is needed. The Reynolds number based on velocity and the hydraulic diameter was varied between 1000 and 2300.

In this work, only sharp bend is considered without the curvature ratio. The Reynolds number ranges from 1000 to 2200, for which the loss coefficient is independent of Re [15]. Therefore, in this case only three parameters have effect on the loss factor.

A number of configurations with different *a*, *b* and *c* were studied using the methodology mentioned above. The bend loss coefficient can be estimated using eq. (12). The method was used before and has a good agreement with the experimental data[15]. The influences of geometric parameters on the bend loss coefficient are discussed below.

3.1 Effect of Aspect Ratio

The effect of aspect ratio on loss coefficient on the condition of a<1 was studied before [15]. In this work the aspect ratio within the range from 1 to 6 was investigated. The present calculations for a=1 are in good agreement with that work. However, a higher aspect ratio channel is often used in heat sink. In the optimized result of Knight [2], the aspect ratio was larger than 1. With a number of repeat calculations, it was found that when aspect ratio increases the loss coefficient decreases, but when aspect ratio is larger than 5, the loss coefficient increases.

3.2 Effect of Width of Fins

The effect of width of fins has also been investigated for 1 to 30 times hydraulic diameters [15]. However the ratio of fin width to channel width (b) less than 1 appeared in heat sink design [2], therefore this range need to be studied. In this work, b ranges from 0.25 to 1.5. When b increases from 0.25 to 1, the loss coefficient decreases rapidly; when b approaches 1, minimum value of loss factor appears; when b continually increases from 1, loss coefficient increases.

3.3 Effect of Turn Clearance

The turn clearance also has effect on pressure drop, which was demonstrated by experiments data [16]. Narrow turn clearance causes high speed of the flow passing through the bend, larger turn clearance incurs a deceleration. Here, the ratio of turn clearance to channel width (c) was used to show the effect on loss coefficient. In this work, c ranges from 0.5 to 6. Fig.(2) shows that when c increases to about 2, the loss coefficient decreases rapidly; when c continues increases from 2 to 6, it changes little. These results have the same trend with experimental results [16, 17]. For serpentine channel heat sink design in this work, the c=2 was fixed for its lower loss factor and little change to heat transfer.

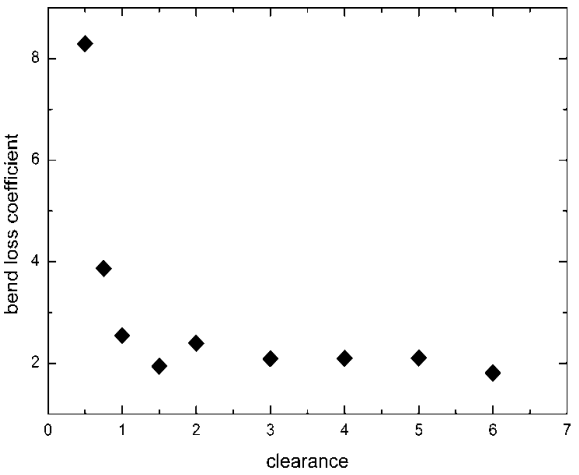


Fig. 2. Bend loss coefficient for different c

3.4 Correlation for the Bend Loss Coefficient

Based on a number of simulations, a correlation is developed for bend loss coefficient as a function of a, b and c for 1000<Re<2300. The proposed correlation is as follow:

$$\xi=8.09\left(1-0.3439a+0.042a^2\right)\left(1-0.3315b+0.1042b^2\right) \tag{16}$$

The correlation given by eq. (16) is valid for the following range of parameters:

$$1000<\mathrm{Re}<2300;\ 1<a<6;\ 0.25<b<1.5$$

4 Optimization with Multi-objective Genetic Algorithms

4.1 Objective Function, Design Parameters and Constraints

The goal of this study is to find the optimal geometric parameters that simultaneously minimize thermal resistance and pressure drop in serpentine channel heat sink. The

length (L) and width (W) of heat sink and the height of base (t_b) are fixed. The objectives are written as following:

$$f_1 = R_{tot}, \quad f_2 = \Delta P \tag{17}$$

where R_{tot} and ΔP are determined by the number of channels (N), width of channel (w_c), height of channel (H) and the velocity of inlet (v). Since the width of heat sink is constant, the sum of width of channels and fins is constant, namely, the equation below should be satisfied:

$$Nw_c + (N+1)w_f = W \tag{18}$$

The constraint is also introduced to insure that the flow should be in laminar regime, namely Re<2300. To obtain the Pareto optimal solutions, a multi-objective genetic algorithm is used based on NSGA-II.

4.2 Results of NSGA-II

To minimize the thermal resistance and the pressure loss of the serpentine channel heat sink, four design parameters including the number of channels (N), width of channel (w_c), height of channel (H) and the velocity of inlet (v) are involved. Design parameters and the range of their variations are listed in Table 1. The parameters of heat sink are shown in Table 2. The optimization was performed to find the optimal solution for the heat sink. The Pareto optimal solutions were obtained by a real-coded NSGA-II with 150 generations and population size of 200 individuals. The Pareto-optimal solutions are shown in Fig. 3, which clearly reveal that these two objectives are conflicting with each other. Any design parameters change that decreases the thermal resistance leads to an increase in the pressure drop and vice versa. Designers can pick up any optimal solution on the Pareto-optimal curve in accordance with the available pressure drop to drive the coolant.

Table 1. The design parameters and the range of their variations

Variables	From	To
Number of channels (-)	4	20
Width of channel (mm)	1	4
Height of channel (mm)	2	5
Velocity of inlet (m/s)	0	2

Table 2. Parameters of heat sink

Length of heat sink (mm)	32	Thermal conductivity of heat sink ($W/m^{-1}K^{-1}$)	160
Width of heat sink (mm)	32	Density of water (kg/m^3)	1000
Thickness of base plate (mm)	1	Water kinematic viscosity (m^2/s)	1e-6
Ambient temperature (K)	293.15	Water Prandtl coefficient (-)	7
Thermal conductivity of water ($W/m^{-1}K^{-1}$)	0.595	Water heat capacity ($Jkg^{-1}K^{-1}$)	4.183e3

4.3 Verification by Numerical Simulation

In the present study, the optimized solutions were verified by numerical simulations. The calculation domain including the flow field and heat sink is same as Fig. 1(c). CFD software was used to calculate the temperature distribution of heat sink and pressure loss in flow field. The flow field was meshed the same way as described in section 3. The solid part of heat sink was also meshed appropriately which was refined at the bends and fins. The boundary conditions for simulation were presented in section 3 with the difference of energy equation concerned. The temperature of ambient and inlet coolant is 293.15K. Uniform heat flux loading on the bottom plate is 6000W/m^2. The top plate of heat is consumed to be insulation. After convergence, the maximum temperature of heat sink can be obtained, and the total thermal resistance can be calculated by the left part of eq. (1). The pressure differences between the inlet and outlet averaged values were also computed.

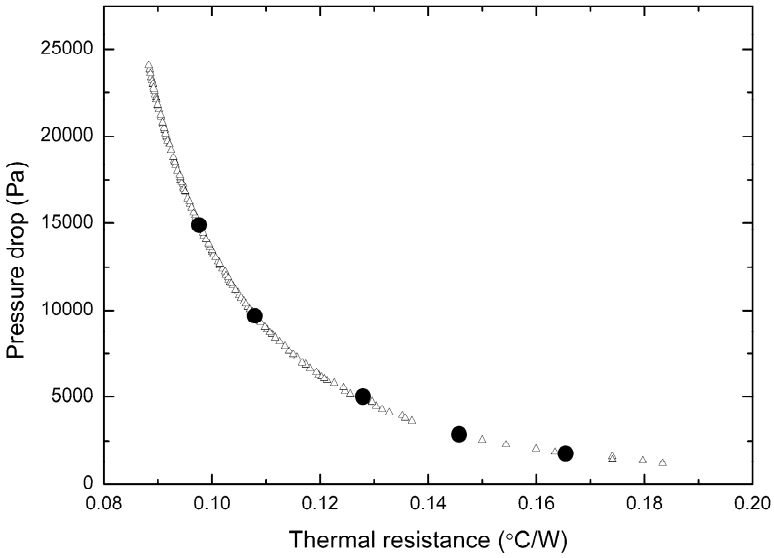


Fig. 3. Pareto-optimal solutions of optimization

The Five representative solutions (with lower values of thermal resistance and pressure drop) on the Pareto front are selected to validate by CFD. The values of objective functions and the corresponding design variables and numerical simulations at these five representative solutions are reported in Table 3.

Table 3. Numerically predicted values

Designs	Variables				Objective function values					
					Model		Numerial analysis		%Error	
	N	w_c	H	v	Rtot	P	Rtot	P	Rtot	P
A	12	2	5	0.7858	0.0976	14894.5	0.1042	15099.3	6.3%	1.3%
B	11	2.2	5	0.6742	0.1079	9678.6	0.1107	10375.2	2.5%	6.7%
C	10	2.6	5	0.4967	0.1279	5026.4	0.1239	5407.9	3.2%	7.1%
D	10	2.8	5	0.3885	0.1457	2869.5	0.1396	3260.8	4.4%	12%
E	9	2.8	5	0.3022	0.1654	1758.0	0.1569	1725.1	5.4%	1.9%

It is clearly shown in Table 3 that the values of variables of selected solutions are reasonable to design a heat sink. The differences between analytical model and numerically simulation are very small for two objective functions, thermal resistance and pressure drop. The error of pressure drop of design D is relatively big than others, because in design D the ratio of width of fin and width of channel is about 0.13, which is out of the range of bend loss coefficient correlation obtained in section 3. The deviation of bend loss coefficient causes the big error of pressure drop.

5 Conclusions

Thermal resistance and pressure drop of serpentine channel heat sink were obtained by using the analysis model. A correlation of pressure loss coefficient for 180 sharp bend has proposed for $1<a<6$ and $0.25<b<2$ with a number of CFD simulations. Next thermal resistance and pressure drop of the heat sink are minimized by multi-objective GAs with fixed length and width of heat sink and the thickness of base plate with four variables viz., the number of channels, channel width, channel height and the inlet velocity. Based on the above analyses, the following conclusions are drawn.

The thermal resistance and pressure drop analysis model have good agreement with the results of numerically calculation.

The bend loss coefficient correlation was obtained by CFD simulations, which was function of the aspect ratio (a), ratio of fins width to channel width (b). The ranges of the two variables a and b were expended.

The thermal resistance and pressure drop of serpentine heat sink are conflicting with each other. Thus, it is import to take into account the pressure drop to design serpentine channel, and designers should pick up the optimal solution along the Pareto front in accordance with available pumping power to drive the coolant.

Acknowledgments. The authors would like to acknowledge the partial supports provided by the National Natural Science Foundation of China (No. 91123023, 61106107 and 51105061), the Fundamental Research Funds for the Central Universities No. E022050205, and the New Century Excellent Talents Program No. NCET-09-0264.

References

[1] Tuckerman, D.B., Pease, R.F.W.: High-Performance Heat Sinking for VLSI. IEEE Electron Device Letters 2, 126–129 (1981)
[2] Knight, R.W., Hall, D.J., Goodling, J.S., Jaeger, R.C.: Heat sink optimization with application to microchannels. IEEE Transactions on Components, Hybrids, and Manufacturing Technology 15, 832–842 (1992)
[3] Perret, C., Boussey, J., Schaeffer, C., Coyaud, M.: Analytic modeling, optimization, and realization of cooling devices in silicon technology. IEEE Trans. Compon. Packag. Technol. 23, 665–672 (2000)
[4] Biswal, L., Chakraborty, S., Som, S.K.: Design and Optimization of Single-Phase Liquid Cooled Microchannel Heat Sink. IEEE Trans. Compon. Packag. Technol. 32, 876–886 (2009)
[5] Gosselin, L., Tye-Gingras, M., Mathieu-Potvin, F.: Review of utilization of genetic algorithms in heat transfer problems. Int. J. Heat Mass Transfer 52, 2169–2188 (2009)
[6] Husain, A., Kim, K.-Y.: Enhanced multi-objective optimization of a microchannel heat sink through evolutionary algorithm coupled with multiple surrogate models. Appl. Therm. Eng. 30, 1683–1691 (2010)
[7] Jeevan, K., Quadir, G.A., Seetharamu, K.N., Azid, I.A., Zainal, Z.A.: Optimization of thermal resistance of stacked micro-channel using genetic algorithms. International Journal of Numerical Methods for Heat & Fluid Flow 15, 27–42 (2005)
[8] Manivannan, S., Devi, S.P., Arumugam, R.: Optimization of flat plate heat sink using genetic algorithm. In: 2011 1st International Conference on Electrical Energy Systems (ICEES), pp. 78–81 (2011)
[9] Peng, C.H., Wu, M.C., Horng, J.T., Lee, C.Y., Fang, C.J., Hung, Y.H.: An optimal approach with genetic algorithm for thermal performance of heat sink/TEC assembly. In: The Tenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems, ITHERM 2006, pp. 458–463 (2006)
[10] Sanaye, S., Hajabdollahi, H.: Thermal-economic multi-objective optimization of plate fin heat exchanger using genetic algorithm. Appl. Energy 87, 1893–1902 (2010)
[11] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197 (2002)
[12] Copeland, D.: Optimization of parallel plate heatsinks for forced convection. In: Sixteenth Annual IEEE Semiconductor Thermal Measurement and Management Symposium, pp. 266–272 (2000)
[13] Pharoah, J.G.: An Efficient Method for Estimating Flow in the Serpentine Channels and Electrodes of PEM Fuel Cells. In: ASME Conference Proceedings, ASME, pp. 547–554 (2006)
[14] Pharoah, J.G.: On the permeability of gas diffusion media used in PEM fuel cells. J. Power Sources 144, 77–82 (2005)
[15] Maharudrayya, S., Jayanti, S., Deshpande, A.P.: Pressure losses in laminar flow through serpentine channels in fuel cell stacks. J. Power Sources 138, 1–13 (2004)
[16] Hirota, M., Fujita, H., Syuhada, A., Araki, S., Yoshida, T., Tanaka, T.: Heat/mass transfer characteristics in two-pass smooth channels with a sharp 180-deg turn. Int. J. Heat Mass Transfer 42, 3757–3770 (1999)
[17] Syuhada, A., Hirota, M., Fujita, H., Araki, S., Yanagida, M., Tanaka, T.: Heat (mass) transfer in serpentine flow passage with rectangular cross-section. Energy Convers. Manage. 42, 1867–1885 (2001)

Towards Non-linear Constraint Estimation for Expensive Optimization

Fabian Gieseke and Oliver Kramer

Department of Computer Science
University of Oldenburg
26111 Oldenburg, Germany

Abstract. Constraints can render a numerical optimization problem much more difficult to address. In many real-world optimization applications, however, such constraints are not explicitly given. Instead, one has access to some kind of a "black-box" that represents the (unknown) constraint function. Recently, we proposed a fast linear constraint estimator that was based on binary search. This paper extends these results by (a) providing an alternative scheme that resorts to the effective use of support vector machines and by (b) addressing the more general task of non-linear decision boundaries. In particular, we make use of active learning strategies from the field of machine learning to select reasonable training points for the recurrent application of the classifier. We compare both constraint estimation schemes on linear and non-linear constraint functions, and depict opportunities and pitfalls concerning the effective integration of such models into a global optimization process.

1 Introduction

Many engineering problems are subject to linear or non-linear constraints. In numerical optimization scenarios, such constraints can make a problem considerably harder to address compared to the standard unconstrained case. Since the evaluation of a constraint function might be very expensive, meta-modeling strategies have already been employed to reduce the number of constraint function calls, similar to objective function meta-modeling that has been widely considered in the past. However, estimating such constraints depicts a challenging task, even if one considers one of the simplest cases like the estimation of a linear (but unknown) hyperplane. Further, the more general (and more complex) case of non-linear constraints has not yet gained much attention up to now.

Recently, we proposed a *linear constraint boundary estimator* (LCBE) [8], which was based on an effective binary search framework. The purpose of this work is to provide a simple yet surprisingly effective constraint estimation model that makes use of state-of-the-art tools from the machine learning field. In particular, we consider an *active learning* [12] strategy that is adapted to the specific needs of *support vector machines* [11,13,15] to iteratively select reasonable points for estimating the desired constraint function within a pre-defined region of interest. The new approach is surprisingly effective and can, in contrast to LCBE, also be applied to non-linear constraints.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 459–468, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Related work on meta-modeling of (non-linear) constraints, but also on general constraint handling for the *covariance matrix adaptation evolution strategy* (CMA-ES) [2,5], is quite rare. One of the first papers on meta-modeling of constraints for evolutionary optimization is the work of Runarsson [10], who employs *nearest neighbor regression* [6] as surrogate model. Recently, Arnold and Jansen [1] introduced a constraint handling method for the $(1 + 1)$-CMA-ES. The idea is based on approximating "the directions of the local normal vectors of the constraint boundaries by accumulating steps that violate the respective constraints, and to then reduce variances of the mutation distribution in those directions" [1]. Hence, this approach iteratively makes use of the information provided by the (evaluated) constraint function to avoid sampling unnecessarily in infeasible regions of the search space. However, no *explicit* meta-model is obtained via this approach. Due to lack of space, we refer to Jin [7] for a comprehensive overview of further work on constraint handling and meta-modeling.

Interestingly, estimating or "learning" a, e. g., linear decision hyperplane in the Euclidean space $\mathbb{R}^d$ can be formalized as active learning [12] problem: Freund *et al.* [4] show that an appropriate selective sampling strategy yields an approximation with error less than $\varepsilon > 0$ by spending $\mathcal{O}(\log(1/\varepsilon))$ hyperplane constraint function calls. Surprisingly, the more general case of estimating a particular class of non-linear decision functions has not yet gained much attention.

This paper is structured as follows: In Section 2, we briefly review the approach for estimating an arbitrary linear constraint hyperplane. This forms the basis for our active learning variant, which is described in Section 3. In Section 4, we compare both approaches on a set of linear constraint hyperplanes, and also show how our extension deals with non-linear problem instances. Conclusions and future research directions are provided in Section 5.

2 Linear Constraint Estimation Revisited

The key idea of the previously proposed linear constraint boundary estimator (LCBE) [8] is based on a simple yet effective binary search scheme that yields d *cutting points*, which lie on or close to the target hyperplane $h \subset \mathbb{R}^d$ (where $h \geq 0$ corresponds to the feasible region and $h < 0$ to the infeasible one). The overall algorithmic framework is depicted in Algorithm 1: Starting with a set of one feasible and d infeasible points (or vice versa), the algorithm performs d binary search steps to identify the desired cutting points. In each step, one considers a pair of a feasible point $\mathbf{x}_F$ and an infeasible point $\mathbf{x}_I$, and iteratively tests the mean of them for validness, see Steps 4–11 of Algorithm 1. The resulting d cutting points form the basis for the computation of the approximation $\hat{h}$ in $\mathbb{R}^d$ in Step 13 (assuming that all cutting points are linearly independent [8]). The identification of the initial set of points can be performed in various kinds of ways. One of them is to sample points from a user-defined probability distribution $\mathcal{P}$ until the computed set of points fulfills the desired properties (we will resort to this strategy in the experimental comparison depicted below).

Input: A linear constraint function h and the number $B_s \in \mathbb{N}$ of allowed binary search steps per cutting point.
Output: An estimation $\hat{h}$ for h.

Initialize set $F = \{\bar{\mathbf{x}}\}$ (feasible point) and set $I = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_d\}$ (d infeasible points).
$S = \{\}$
for $i = 1$ to d **do**
  $\mathbf{x}_F := \bar{\mathbf{x}}$ and $\mathbf{x}_I := \mathbf{x}'_i$
  $k = 0$
  while $k < B_s$ **do**
    $\mathbf{z} = \frac{(\mathbf{x}_F + \mathbf{x}_I)}{2}$
    if $\langle \mathbf{w}, \mathbf{z} \rangle + b \geq 0$ **then** $\mathbf{x}_F = \mathbf{z}$ **else** $\mathbf{x}_I = \mathbf{z}$
    $k = k + 1$
  end while
  $S = S \cup \{\frac{(\mathbf{x}_F + \mathbf{x}_I)}{2}\}$
end for
Compute approximation $\hat{h}$ based on the set S (see text).
return

Algorithm 1. Linear constraint boundary estimator (LCBE) [8]

3 An Active Large Margin Constraint Estimator

We will now develop a more general constraint estimation model that is based on support vector machines in combination with adapted active sampling strategies.

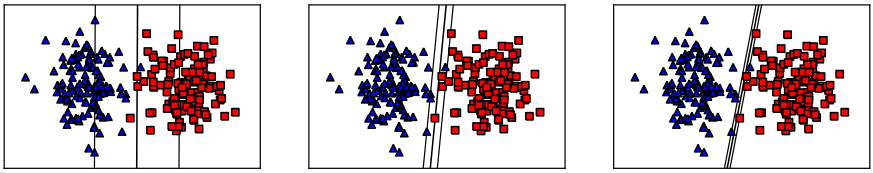
3.1 Support Vector Machines

The idea of a support vector machine is to learn a decision boundary that allows to separate the members of two classes. The optimal linear hyperplane maintains a large distance of the closest patterns, which is also known as *margin* [11,15]. The two classes are represented via a labeled *training set* $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \{-1, +1\}$. Given the separable case (i.e., both classes can be separated by a linear hyperplane), one obtains the following optimization task to be solved for *hard-margin support vector machines* [11,13]:

$$\begin{aligned} \underset{\mathbf{w} \in \mathbb{R}^d,\, b \in \mathbb{R}}{\text{minimize}}\ & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t. } & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \end{aligned} \tag{1}$$

Here, the objective corresponds to maximizing the margin to the given patterns. The concept of hard-margin support vector machines is based on the assumption that the training set is linearly separable. To cope with non-separable training sets, one usually relaxes the constraints by allowing patterns to lie within the margin (or even on the wrong side of the hyperplane). More precisely, so-called *slack variables* $\xi_1, \ldots, \xi_n \in \mathbb{R}$ are introduced to impose

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,\ \xi_i \geq 0 \tag{2}$$



(a) $C = 0.001$ (b) $C = 1$ (c) $C = 1,000$

Fig. 1. The blue triangles depict negative and the red squares positive training instances. The decision hyperplane and the margin are indicated by black lines.

for $i = 1, \ldots, n$. This leads to the following optimization task [11,13]:

$$\underset{\mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R},\, \boldsymbol{\xi}\in\mathbb{R}^n}{\text{minimize}} \; \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \qquad (3)$$
$$\text{s.t. } y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

The first term of the objective corresponds to maximizing the margin and the second one captures the violation of the (previously) strict constraints. The parameter $C \in \mathbb{R}^+$ determines the trade-off between both aims, see Figure 1 for an illustration. Non-linear decision functions are obtained via the use of so-called *kernel functions* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [11,13]. Well-known candidates are, for instance, the *linear kernel* with $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j\rangle$ and the *radial basis function* (RBF) *kernel* with $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ and associated *kernel width* $\gamma \in \mathbb{R}^+$. Such a kernel function gives rise to a *kernel matrix* $\mathbf{K} \in \mathbb{R}^{m\times m}$ with entries $[\mathbf{K}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$; a defining property of a kernel function is the assumption that this matrix must be positive semidefinite.

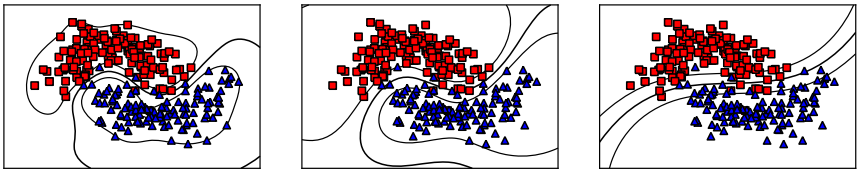
An artificial toy example of a support vector machine model that is based on an RBF kernel is given in Figure 2. Note that, from an optimization point of view, one obtains a *quadratic programming* [3] problem of the form

$$\underset{\boldsymbol{\beta}\in[0,C]^n}{\text{maximize}} \; \sum_{i=1}^{n}\beta_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \qquad (4)$$
$$\text{s.t. } \sum_{i=1}^{n}\beta_i y_i = 0$$

which can be solved efficiently in $\mathcal{O}(n^3)$ time [11,13].

3.2 An Iterative Selection Strategy for Constraint Estimation

Support vector machines depict powerful tools for various learning tasks given a *fixed* training set. However, for the meta-modeling purposes depicted above, one has to generate appropriate training samples that reflect the (unknown) constraint boundary. A natural approach is to select these instances uniformly



(a) $\gamma = 4.0$ (b) $\gamma = 1.0$ (c) $\gamma = 0.25$

Fig. 2. The kernel width $\gamma \in \mathbb{R}^+$ of the RBF kernel is decreased from left to right. Blue triangles depict negative and red squares positive training instances.

at random. It is well known, however, that this approach requires a large amount of sample points to achieve a satisfying accuracy.

A more sophisticated one is to iteratively sample these instances, as we will show next. Note that, from a machine learning point of view, the estimation of constraints can be seen as *active learning* [12] scenario, where a huge amount of unlabeled instances is given and where the labeled instances can be obtained via an evaluation of the constraint function. In the literature, several approaches have been proposed [14]; due to lack of space, we refer to Settles [12] for a comprehensive overview. Our setting is related to such active learning scenarios. However, there are two important differences one can take advantage of:

1. First, assuming a noiseless constraint function, the patterns are separable (i. e., we do not have overlapping classes) by some non-linear mapping (at least, by the constraint function itself).
2. Second, one can sample an arbitrary number of unlabeled patterns that are close to the current hyperplane model (without any additional cost with respect to the evaluation of the true constraint function).

Thus, aiming at the effective use of support vector machines, we can restrict our models to the hard-margin case, assuming that one employs a sufficiently rich kernel function that can reflect the shape of the constraint function g. The algorithmic framework for estimating constraints is shown in Algorithm 2: The goal is to estimate the true constraint function g in a sampling region induced by an input distribution $\mathcal{P}$. Starting with an initial labeled training set of size m that contains both feasible and infeasible points, the approach iteratively (1) computes a hard-margin support vector machine model, (2) generates u unlabeled points according to $\mathcal{P}$, and (3) labels the point that is closest to the current decision surface of the support vector machine model. This process is repeated until a pre-defined budget of constraint calls is exhausted.

Note that the training of the model in each iteration is based on *all* points generated so far, and that the non-fixed parameters are tuned via leave-one-out cross validation [6]. This ensures that one takes advantage of the rare labeled points as well as possible. The overall scheme is driven by the user-defined input probability distribution $\mathcal{P}$ that defines the "region of interest". Without any such assumption, arbitrary non-linear constraint surfaces cannot be handled.

Input: Constraint function $g : \mathbb{R}^d \to \{-1, +1\}$, parameters $m, u, B \in \mathbb{N}$, and a probability distribution $\mathcal{P}$.
Output: An approximation $\hat{g}$ of g in the sampling region induced by $\mathcal{P}$.

1: Select and label an appropriate initial training set T of size m that contains both feasible and infeasible training samples (see text).
2: $i = 0$
3: **repeat**
4: Compute hard-margin support vector machine model $h(\cdot) = \langle \mathbf{w}, \cdot \rangle + b$ based on T (using leave-one-out cross validation for tuning involved kernel parameters).
5: Generate a set $T_u = \{\mathbf{z}_1, \ldots, \mathbf{z}_u\} \subset \mathbb{R}^d$ of u unlabeled points that are sampled according to $\mathcal{P}$.
6: Let $\mathbf{z}_\kappa \in T_u$ be the point that minimizes $d(\mathbf{z}_j, h)$ for $j = 1, \ldots, u$.
7: $T = T \cup \{(\mathbf{z}_\kappa, g(\mathbf{z}_\kappa))\}$
8: $i = i + 1$
9: **until** $m + i > B$

Algorithm 2. Active Hard-Margin Constraint Estimation (AHMCE)

4 Experimental Evaluation

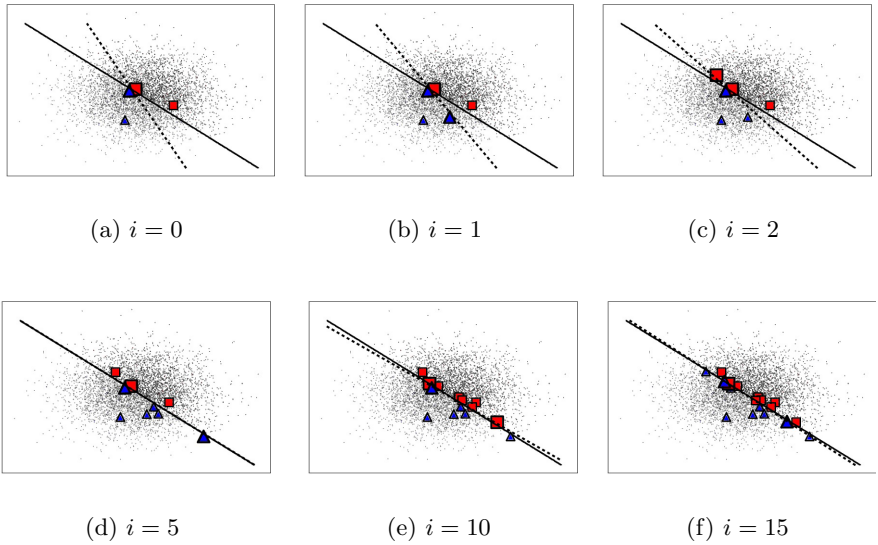
In the following, we will analyze the above meta-models given both linear and non-linear settings, and will sketch an exemplary use of such models for global optimization frameworks. Our implementations are based on `Python 2.6` including the `scikit-learn` package. For the support vector machine implementation, we fix C to $10,000$ to enforce hard-margin models.[1]

4.1 Constraint Estimation Analysis

We start by comparing the LCBE method with our new AHMCE framework on a simple hyperplane constraint in $\mathbb{R}^d$, followed by non-linear scenarios.

Linear Constraints. To compare both schemes for linear settings, we define $g_1(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ with $\mathbf{w} = (1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^d$, $b = 0$, and $\text{sgn}(x) = 1$ for $x \geq 0$ and $\text{sgn}(x) = -1$ otherwise, and fix $\mathcal{P} = \mathcal{N}(\mathbf{0}, \mathbf{I})$. For LCBE, we limit the number B_s of binary search steps per cutting point to 5, and set B for AHMCE to the number of constraint calls used by LCBE. Further, we fix $m = 4$ and $u = 1,000$ for AHMCE. Estimating this constraint function gets more and more difficult with increasing dimension of the search space. To illustrate this issue and to compare the accuracy of both models, we vary the dimension d from 2 to 15. The approximations obtained via both schemes are shown Figures 4 (a) and (b) for $d = 2$; the test errors[2] for the remaining dimensions are shown in

[1] Since the induced task is separable via a non-linear function, hard-margin support vector machines depict appropriate candidates to represent the constraint function.
[2] Which are obtained on an independent test set of $100,000$ points.



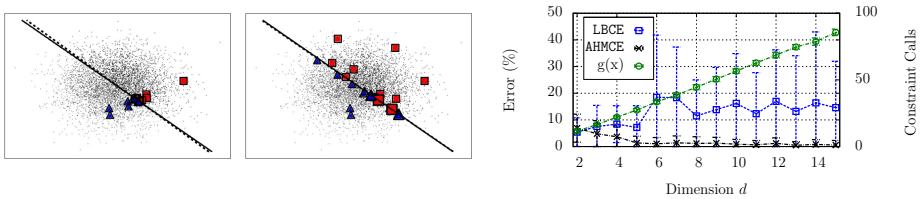
(a) $i = 0$ (b) $i = 1$ (c) $i = 2$

(d) $i = 5$ (e) $i = 10$ (f) $i = 15$

Fig. 3. Algorithm 2 with $\mathcal{P} = \mathcal{N}(\mathbf{0}, \mathbf{I})$: The small points indicate the underlying distribution $\mathcal{P}$; larger red squares and blue triangles depict the points that have already been labeled. Starting with an initial set of both feasible and infeasible solutions ($i = 0$), the approach iteratively labels a pattern that is close to the approximation (dashed line) of the true constraint function g (solid line).

Figure 4 (c). It can be seen that both approaches can effectively estimate the constraint hyperplane for $d = 2$. For higher dimensions, however, the AHMCE model is clearly superior. Two issues should be pointed out: First, the approximation quality of LCBE can naturally be improved by increasing the number of allowed binary search steps. Second, since the hard-margin support vector model is based on quadratic programming, one would need to increase the accuracy for the intermediate models in case a higher overall accuarcy is desired.

Non-linear Constraints. The new constraint estimator can also, in contrast to LCBE, deal with non-linear settings due to the concept of kernel functions. To illustrate these capabilities, we consider two non-linear constraint functions for $d = 2$, namely a "sinus example" defined via $g_2(\mathbf{x}) = \text{sgn}(\sin(2x_0) - x_1)$ and a "XOR example" denoted by g_3. The approximations computed by the AHMCE approach (with $m = 25$, $B = 100$, and an RBF kernel with $\gamma \in \{2^{-10}, \ldots, 2^{10}\}$) for these two examples are shown in Figure 5, given two different input distributions. Naturally, and in contrast to the linear setting, one has to make a compromise between (a) diversity and (b) accuracy of the model. In our setting, both the number m of initial (random) starting points as well as the input distribution $\mathcal{P}$ determine this compromise. While $\mathcal{P} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ induces a more detailed model around the origin, $\mathcal{P} = \mathcal{N}(\mathbf{0}, 2\mathbf{I})$ enforces more diversity and, hence, can better capture the global structure of the functions.



(a) LCBE (b) AHMCE (c)

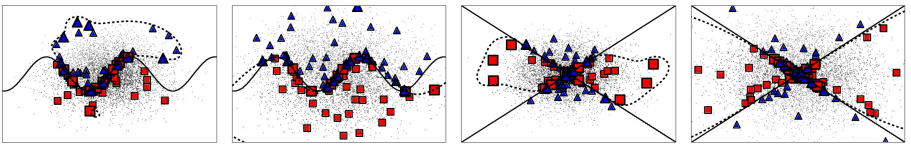
Fig. 4. Estimation of a linear hyperplane: Figures (a) and (b) show the outcome for the two-dimensional case. Figure (c) depicts the average test errors over 25 runs for varying dimensions $d = 2, \ldots, 15$. LCBE and AHMCE resort to the same amount of constraint function calls (green curve). Clearly, AHMCE yields much better models (with smaller errors) as LCBE spending the same amount of constraint function calls.

4.2 Application: Meta-modeling in Optimization

The meta-models described above can be employed for various applications. One of them is constrained optimization, and we briefly sketch the opportunities and pitfalls for such scenarios. For the sake of demonstration, and to facilitate the setup, we make use of a standard CMA-ES [2,5] optimization process for the fitness function $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{x}$ subject to the constraints defined above (with $d = 2$). The CMA-ES is initialized with standard parameters using $\mathbf{x}_s = (10, 0)$, $\mathbf{x}_s = (1, 1)$, and $\mathbf{x}_s = (1, 1)$ as initial candidate solutions for g_1, g_2, and g_3, respectively, initial step size $\sigma = 0.5$, population size $\lambda = 4 + 3\log(d) = 7$, and target fitness $f_t = 0.001$. To integrate the meta-models, we follow our previous work [8] and train them off-line, beforehand; the CMA-ES then resorts to this surrogate instead of the constraint function. Both meta-models are compared with a standard death penalty enhanced CMA-ES framework.[3]

The results are shown in Table 1. For the linear constraint, both models can successfully be employed to significantly reduce the amount of needed constraint functions calls (from 334 to less than 100). However, for the non-linear constraints, the LCBE does not always yield the optimal solution (average errors of 0.06 and 0.01, respectively). Further, the convergence behavior is worse compared to its competitors (more than 300 fitness evaluations), which stems from bad search directions induced by inappropriate approximations. In contrast, AHMCE can successfully detect the non-linear structures and yields approximation accuracies that are sufficient for a successful replacement of the true constraint functions (about 100 constraint function calls are saved for both problem instances). Still, in case of a bad approximation, the convergence behavior can also become worse, as it is the case for the g_3 constraint (the amount of fitness evaluations is doubled compared to death penalty).

[3] Death penalty resorts to the true constraint function and samples, in each generation, new candidate solutions again and again until sufficient valid candidates are found.



(a) $\mathcal{P} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ (b) $\mathcal{P} = \mathcal{N}(\mathbf{0}, 2\mathbf{I})$ (c) $\mathcal{P} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ (d) $\mathcal{P} = \mathcal{N}(\mathbf{0}, 2\mathbf{I})$

Fig. 5. Two non-linear constraint examples with two input distributions each. The AHMCE scheme can successfully capture the structure of these constraints in the sample region induced by $\mathcal{P}$.

Table 1. Use of the different meta-models for a simple CMA-ES based optimization process induced by the two-dimensional sphere function $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{x}$ subject to a linear constraint $g_1(\mathbf{x}) \leq 0$ (hyperplane) and two non-linear constraints $g_2(\mathbf{x}) \leq 0$ (sinus) and $g_3(\mathbf{x}) \leq 0$ (xor). Mean and one standard deviation for the best reached fitness value (best), the number of needed fitness function calls (f-calls), and the number of needed constraint function calls (c-calls) are reported.

	Death Penalty			LCBE			AHMCE		
	best	f-calls	c-calls	best	f-calls	c-calls	best	f-calls	c-calls
$g_1(\mathbf{x})$	0.00 ± 0.00	225 ± 42	334 ± 55	0.00 ± 0.00	255 ± 22	98 ± 6	0.00 ± 0.00	248 ± 26	95 ± 7
$g_2(\mathbf{x})$	0.00 ± 0.00	140 ± 31	221 ± 43	0.06 ± 0.13	338 ± 377	138 ± 67	0.00 ± 0.00	141 ± 22	144 ± 7
$g_3(\mathbf{x})$	0.00 ± 0.00	125 ± 20	230 ± 27	0.01 ± 0.03	409 ± 406	142 ± 66	0.00 ± 0.00	215 ± 230	148 ± 39

5 Conclusions and Future Work

The approximation of constraint functions is a difficult task. This work extends our previous work [8] on linear constraint estimation. Our approach is based on active learning strategies for hard-margin support vector machines, and we successfully make use of the specific properties of the task at hand to reduce the amount of needed constraint function calls. The new approach achieves a better approximation accuracy compared to LCBE for linear constraint settings, given the same budget of constraint function calls. Further, it can be easily be extended to the non-linear case via the use of kernel functions. As shown in the experimental evaluation, linear constraints can successfully be approximated; non-linear ones, however, depict challenging tasks, even for the two-dimensional case. However, depending on the particular problem instance, constraint function calls can be saved in the context of expensive optimization settings.

We plan to extend these results in future. First, estimating non-linear constraints is very challenging and the question arises if meaningful sampling strategies for arbitrary non-linear constraint settings can be derived at all. Here, a promising research direction is the extension of recently developed *locally linear support vector machines* [9] to the active learning settings considered in this work; such models could still capture constraint functions induced by multiple

linear constraints, while taking advantage of the fact that (partially) linear constraints can be much better detected compared to arbitrary non-linear ones. Second, the integration of such models into global optimization frameworks requires sophisticated meta-model managing systems that, e. g., keep track of the meta-model quality and that invoke retraining phases in case the quality is bad (again, estimating the quality is challenging due to the lack of labeled data).

Acknowledgements. The authors would like to thank the anonymous reviewers for their useful comments and suggestions. This work has been supported by funds of the *Deutsche Forschungsgemeinschaft* (grant KR 3695/2-1).

References

1. Arnold, D.V., Hansen, N.: A (1+1)-CMA-ES for constrained optimisation. In: GECCO, pp. 297–304 (2012)
2. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proceedings of the IEEE Congress on Evolutionary Computation – CEC 2005, vol. 2, pp. 1769–1776 (2005)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge Uni. Press (2004)
4. Freund, Y., Seung, H., Shamir, E., Tishby, N.: Selective sampling using the query by committee algorithm. Machine Learning 28(2-3), 133–168 (1997)
5. Hansen, N.: The CMA evolution strategy: a comparing review. In: Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms, pp. 75–102. Springer (2006)
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer (2009)
7. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation 1(2), 61–70 (2011)
8. Kramer, O., Barthelmes, A., Rudolph, G.: Surrogate Constraint Functions for CMA Evolution Strategies. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 169–176. Springer, Heidelberg (2009)
9. Ladický, L., Torr, P.H.S.: Locally linear support vector machines. In: Proceedings of the 28th International Conference on Machine Learning, pp. 985–992 (2011)
10. Runarsson, T.P.: Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 401–410. Springer, Heidelberg (2004)
11. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2001)
12. Settles, B.: Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2012)
13. Steinwart, I., Christmann, A.: Support Vector Machines. Springer, New York (2008)
14. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research 2, 45–66 (2002)
15. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)

Repair Methods for Box Constraints Revisited

Simon Wessing

Fakultät für Informatik, Technische Universität Dortmund, Germany
simon.wessing@tu-dortmund.de

Abstract. Box constraints are possibly the simplest kind of constraints one could think of in real-valued optimization, because it is trivial to detect and repair any violation of them. But so far, the topic has only received marginal attention in the literature compared to the more general formulations, although it is a frequent use case. It is experimentally shown here that different repair methods can have a huge impact on the optimizer's performance when using the covariance matrix self-adaptation evolution strategy (CMSA-ES). Also, two novel repair methods, specially designed for this algorithm, sometimes outperform the traditional ones.

Keywords: box constraints, repair method, Baldwin, Lamarck.

1 Introduction

In this paper we will deal with single-objective optimization problems given as $f : \mathcal{F} \subset \mathbb{R}^n \to \mathbb{R}$. A solution $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathcal{F}$ is assigned an objective value $f(\boldsymbol{x})$, which is to be minimized. The feasible region is simply defined here as $\mathcal{F} = [\ell, u]^n$, meaning that for each x_i, there is a lower and upper limit ℓ, u, so that $\ell \leq x_i \leq u$. These limits are a special case of linear constraints, where the hyperplanes defining them are axis-aligned. $\mathcal{F}$, the intersection of all the feasible half spaces, is a hypercuboid, hence the name box constraint.

Apparently, box constraints or even linear constraints are usually not treated specially in the literature concerning constraint handling in evolutionary algorithms (EA). Penalty and resampling methods seem to be most common (see Coello Coello [1] for an overview). To the author's best knowledge, the only detailed analysis on repair functions for box constraints was done by Arabas et al. [2]. They also experiment with several other constraint handling methods and apply them to differential evolution on the CEC 2005 benchmark set [3].

Let us depict repair functions as mappings $T : \mathbb{R}^n \to \mathcal{F}$. There are at least three candidates for T that can be decomposed into n one-dimensional repair functions $T_1(x_1), \ldots, T_n(x_n)$, where $T_i : \mathbb{R} \to [\ell, u]$. Projection is probably the simplest one, because any variable that violates a constraint is just set to the value of this constraint:

$$T_i(x_i) = \begin{cases} x_i & \ell \leq x_i \leq u\,, \\ u & x_i > u\,, \\ \ell & x_i < \ell\,. \end{cases}$$

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 469–478, 2013.
© Springer-Verlag Berlin Heidelberg 2013

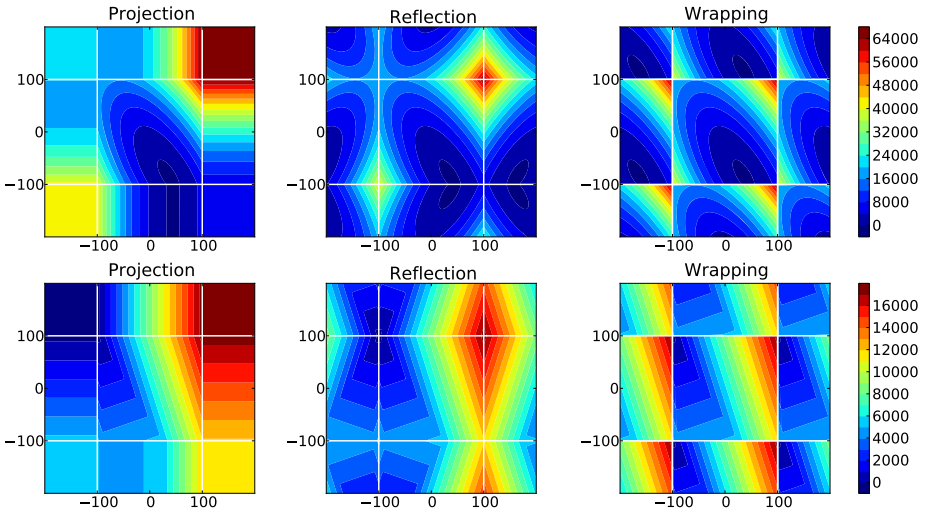


Fig. 1. The two unimodal problems F_2 (top) and F_5 (bottom). The boundaries are marked by white lines. Depending on the repair method, there are different looking "virtual" landscapes outside the boundaries.

This approach, which is promising when the optimum lies on a constraint, seems to be frequently used [4–6]. The other two can be defined recursively. The reflection approach is given by

$$T_i(x_i) = \begin{cases} x_i & \ell \leq x_i \leq u\,, \\ T_i(u + (u - x_i)) & x_i > u\,, \\ T_i(\ell + (\ell - x_i)) & x_i < \ell\,, \end{cases}$$

while Purchla et al. [7] also mention the possibility of treating the search space as a torus. They call this approach wrapping:

$$T_i(x_i) = \begin{cases} x_i & \ell \leq x_i \leq u\,, \\ T_i(x_i - (u - \ell)) & x_i > u\,, \\ T_i(x_i + (u - \ell)) & x_i < \ell\,. \end{cases}$$

Projection and reflection are more versatile than wrapping, because they are also defined when either $\ell = -\infty$ or $u = \infty$. On the other hand, projection introduces a bias towards the boundary, which wrapping and reflection do not. These three alone already seem quite diverse, but later we will see that there are even more aspects to repair methods.

Figure 1 shows on two unimodal problems, how these mappings de facto shape a landscape outside the feasible region. The figure was created by sampling the infeasible regions, repairing the solutions with the respective method, and then evaluating the repaired solution. Both problems are taken from the CEC 2005

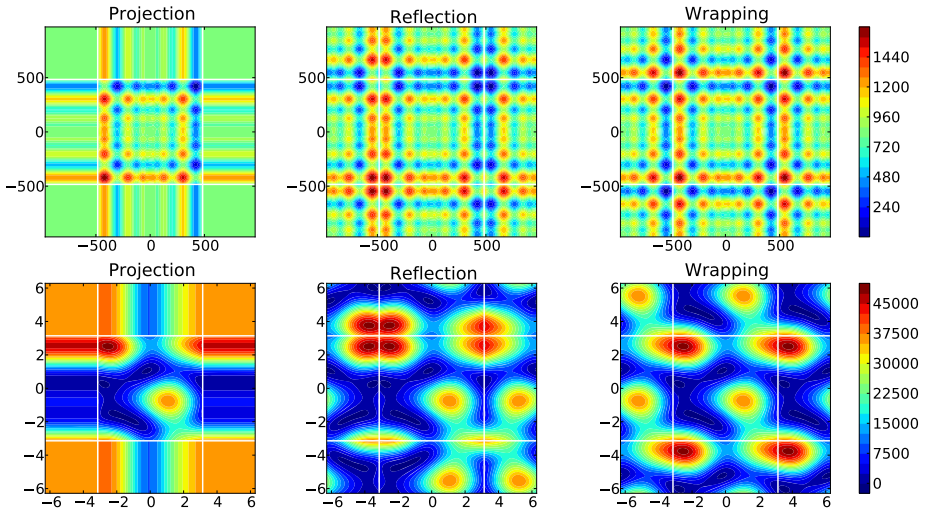


Fig. 2. The two multimodal problems Schwefel (top) and F_{12} (bottom)

special session on real-parameter optimization [3]. F_2 is a shifted double sum problem. Its optimum lies near, but not directly on the boundary. F_5, on the other hand, has its optimum lying in a corner for $n = 2$, while for higher dimensions it is on the edge of the feasible space. Figure 2 shows two multimodal problems. The first one is the generalization of Schwefel's problem 2.26 [8] to n dimensions, $f(\boldsymbol{x}) = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right)$. This problem is highly multimodal. The boundaries are set to $\ell = -(7\pi)^2$ and $u = (7\pi)^2$ to make it wrap around, i. e. the tiled, unconstrained landscape of the problem has no cliffs. It is also deceptive, because large local optima are located far away from each other and the global one. F_{12} is the instance of Fletcher and Powell's problem [9] that also appears in the CEC 2005 competition [3]. It, too, has the special property of wrapping around.

This work focuses on a smaller set of problems and only a subset of the constraint handling methods in [2], but instead adds the question to the investigation, if repairs should be passed on to the offspring. In biology, this concept is known as Lamarckian inheritance and describes the belief that an individual can pass on characteristics that it acquired during its lifetime to its offspring. The alternative theory, which is in accordance with Darwin's theory of natural selection, is the theory of Baldwinian evolution.

In the optimization context, repair functions can be seen as a special case of a local search component added to the EA [10, p. 214]. The repaired solution can either replace the original one (Lamarck), or only the repaired version's fitness is assigned to the original solution (Baldwin). The latter approach is used for example by Hansen et al. [6]. In the software, this corresponds to either implementing the repair as part of the individual (Lamarck) or the problem (Baldwin). Figure 3 shows where the repaired solutions would be located in

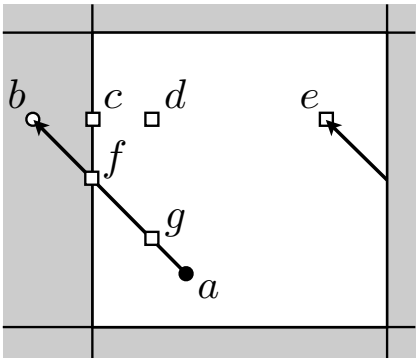


Fig. 3. Illustration of the different possible repair methods. From the feasible solution (a) an infeasible one (b) is created by variation. This solution can be repaired by projection (c), reflection (d), wrapping (e), projection onto the point of intersection (f), or reflection along the variation direction (g).

an example situation. In the case of Baldwinian inheritance, the constraints would be hidden (to the individual) and point $\boldsymbol{b}$ would always seem feasible, but the objective value nonetheless depend on the repaired version. Otherwise, the mapping also influences the following search steps. Imagine, for example, that solution $\boldsymbol{e}$ survives the selection step in the case of Lamarckian inheritance. Assuming intermediate recombination, the search would be dragged away from the boundary, because the next offspring would be created around the center of the then widespread population. This is probably an undesired behavior that should be avoided.

We also want to test two new repair functions ($\boldsymbol{f}$ and $\boldsymbol{g}$ in Fig. 3) that take the mutation direction into account. The approach is similar to Michalewicz' GENOCOP III [11] in that it needs a feasible reference point $\boldsymbol{a}$ and aims to find the intersection of the boundary with the line defined by $\boldsymbol{a}$ and the infeasible point $\boldsymbol{b}$. For a reasonable integration into the variation operator, $\boldsymbol{a}$ should be chosen as the intermediate solution that results from recombination. I. e., for the covariance matrix self-adaptation evolution strategy (CMSA-ES) by Beyer and Sendhoff [12], $\boldsymbol{a}$ would be the population center. This requires that recombination produces a feasible solution, which is guaranteed e. g. for convex $\mathcal{F}$ and operators that generate solutions in the convex hull of the parents.

In the case of box constraints, computing the intersection analytically is easy. Let $x_i = c$ be the hyperplane of the violated constraint and $h : \boldsymbol{x} = \boldsymbol{b} + s(\boldsymbol{a} - \boldsymbol{b})$ the mentioned line. Equating the two yields the scale factor $s = (c - b_i)/(a_i - b_i)$, whose substitution back into h yields the intersection point. If more than one constraint is violated, there exists (in the non-degenerate case) exactly one feasible intersection between $\boldsymbol{a}$ and $\boldsymbol{b}$. Optionally, instead of using the intersection as the repaired version (IP), it is also possible to combine the approach with reflection (IR) or wrapping along h (only the former is tried here).

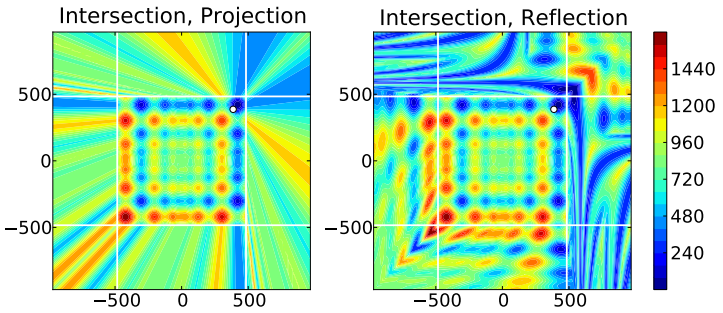


Fig. 4. The two new repair methods on Schwefel's function. The landscape is heavily dependent on the reference point (white dot).

The approach cannot be plotted as easy as the three simple ones, because of its dependency on the reference point $\boldsymbol{a}$. So, the landscape changes every generation. Figure 4 shows an example situation on Schwefel's function. Also, as the reference point has to be feasible, it is not obvious how to combine the mapping with Baldwinian inheritance. Therefore, it is only tested with Lamarckian inheritance in Sec. 2.

2 Experiments

Research Question: Which combination of mapping and inheritance yields the best results? Is there a benefit in using the reference point dependent methods?

Pre-experimental Planning: The CMSA-ES seems to be an appropriate choice for this basic research, because it represents a reasonable compromise between performance and simplicity. There are also two arguments for IP and IR to be tested especially with this algorithm: Firstly, as the reference point $\boldsymbol{a}$ changes every generation, it seems advisable to use an EA with comma selection to not compare individuals that have been repaired differently. Secondly, CMSA-ES uses self-adaptation to adjust the mutation strength σ. This means that a separate σ_i is stored for each individual i, which makes it possible to also scale down the σ_i after an individual is repaired. This is done by setting $\sigma_i := \sigma_i(\|\boldsymbol{f} - \boldsymbol{a}\|)/(\|\boldsymbol{b} - \boldsymbol{a}\|)$ in the case of IP and $\sigma_i := \sigma_i(\|\boldsymbol{g} - \boldsymbol{a}\|)/(\|\boldsymbol{b} - \boldsymbol{a}\|)$ for IR. As it is guaranteed by construction that $\boldsymbol{f}$ and $\boldsymbol{g}$ lie closer to $\boldsymbol{a}$ than $\boldsymbol{b}$, this scaling can only decrease σ_i. In preliminary runs, IP and IR without the scaling could not compete with the other repair methods because of diverging step sizes.

Task: The task is to compare the repair methods regarding their performance. As performance measure, the expected running time [13]

$$\mathrm{ERT} = \mathrm{RT_S} + \frac{1 - \widehat{p}_s}{\widehat{p}_s} \cdot \mathrm{RT_{US}}\,,$$

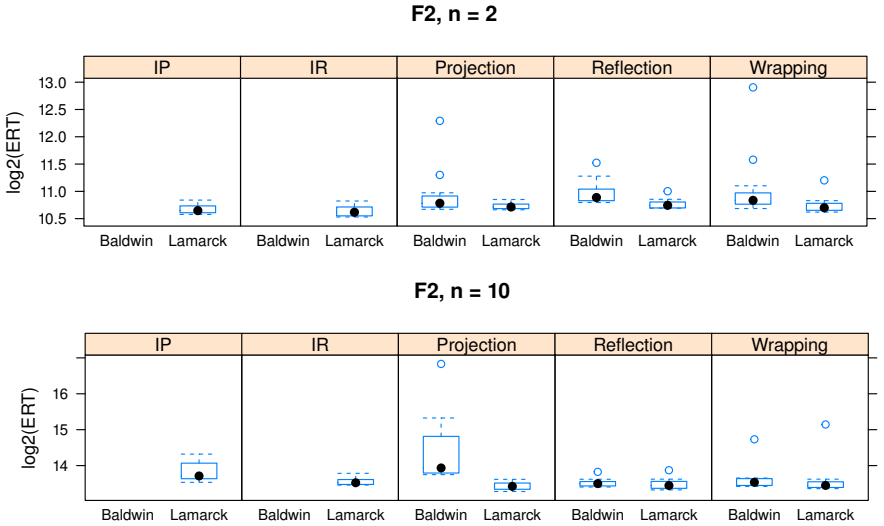


Fig. 5. ERT results on F_2 in two (top) and ten dimensions (bottom)

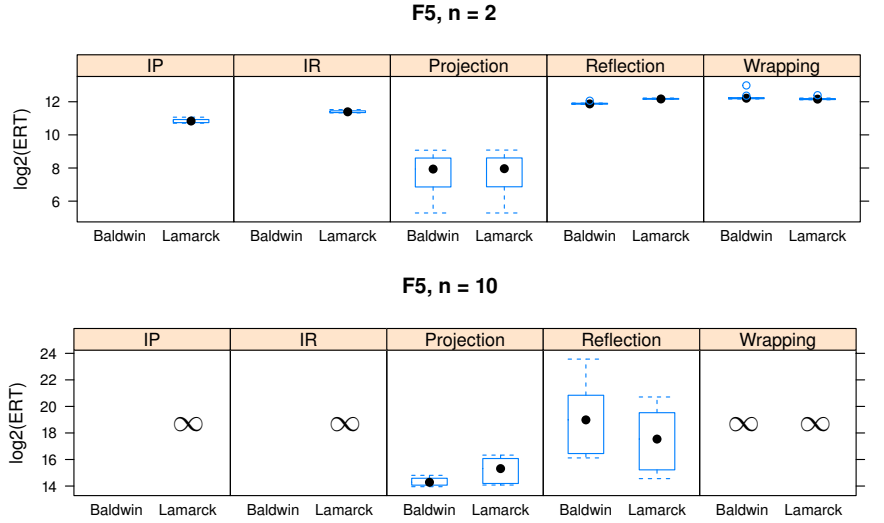


Fig. 6. ERT results on F_5. The ∞ symbols mark configurations with $\widehat{p}_s = 0$.

will be used. $\mathrm{RT_S}$ and $\mathrm{RT_{US}}$ are the average number of function evaluations for successful and unsuccessful runs, while $\widehat{p}_s$ is the estimated success probability of finding the global optimum. In case of $\widehat{p}_s = 0$ we define ERT $= \infty$. For the multimodal problems, figures will focus on $\widehat{p}_s$ itself for better visualization. A run is considered successful when it finds an objective value within 10^{-6} of the global optimum.

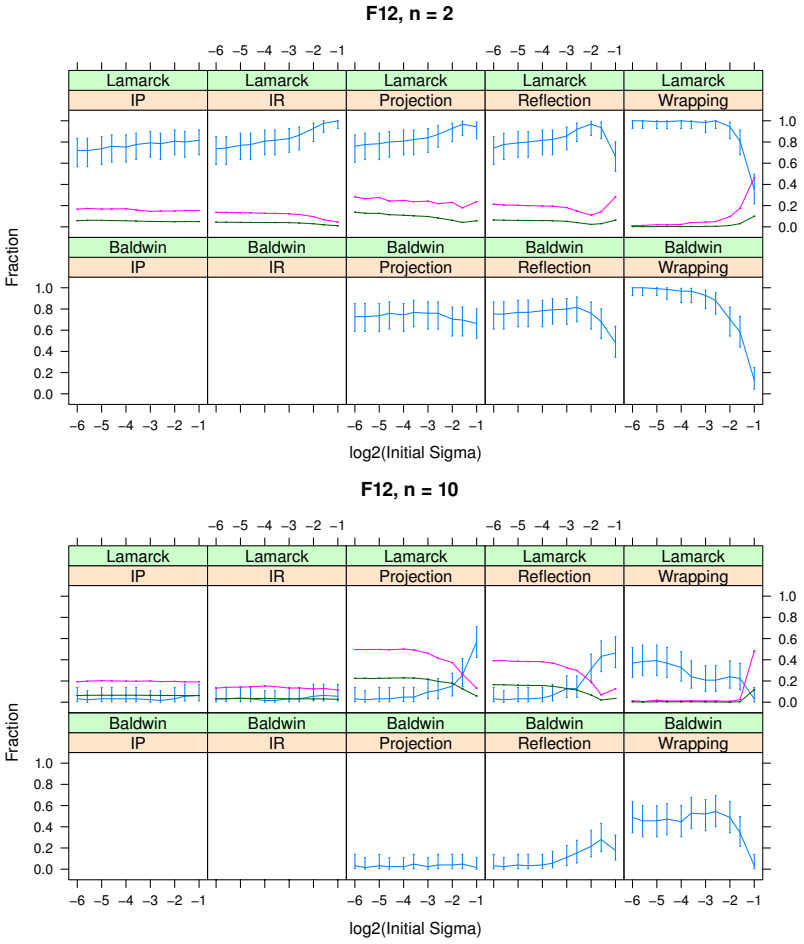


Fig. 7. The success rate $\widehat{p}_s$ (blue), the fraction of repaired offspring (pink), and the repaired offspring that also survived the selection step (green). For Baldwinian inheritance, the latter two measures and IP and IR are not applicable.

Setup: The experiment is carried out on the mentioned problems for $n = 2$ and $n = 10$. A $(15\,,60)$-CMSA-ES is applied to each problem. Besides the repair methods described in Sec. 1, the initial mutation strength σ_{init} is introduced as a factor and the tested levels are $(u-\ell)/k$, $k \in \{2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64\}$. Each configuration is repeated 125 times with the start point drawn uniformly from the feasible region. The maximal number of function evaluations for each run is set to $n \cdot 10^4$. The EA can abort earlier, if a stopping criterion triggers. This is the case when all objective values of the population are less than 10^{-12} away from each other or when all $\sigma_i > 3(u - \ell)$.

Results: Figure 5 shows the ERT results on the problem F_2. On this problem, all the outliers with bad performance are due to the highest step sizes. This

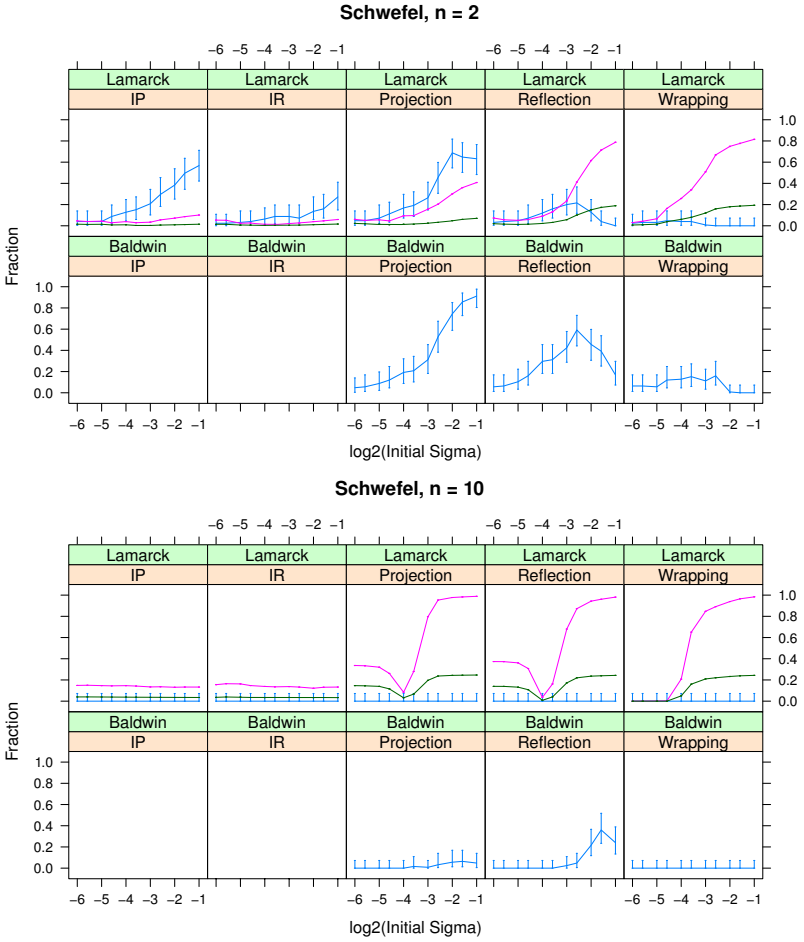


Fig. 8. The success rate $\widehat{p}_s$ (blue), the fraction of repaired offspring (pink), and the repaired offspring that also survived the selection step (green)

also holds for F_5 in two dimensions (see Figure 6), but not in ten dimensions (where the situation is reversed). Of course, there is a clear advantage for projection in two dimensions, because the optimum lies in a corner. In Figs. 7 and 8, $\widehat{p}_s$ is displayed together with the fraction of repaired offspring and the fraction of repaired offspring that survived the selection step. All three are plotted against σ_{init}. The error bars shown there are 95% confidence intervals computed with exact binomial tests. For the fractions of repaired offspring, the error bars are hardly visible, because the number of trials corresponds to the number of function evaluations (while for $\widehat{p}_s$ it is only the number of repeats).

Discussion: The number of repaired offspring that survive selection seems to be roughly proportional to the number of repaired offspring. This may be due to comma selection, which enforces the replacement of all parents, regardless of

the offsprings' objective values. On F_2 and F_5, the fraction of repaired offspring is always increasing with σ_{init} (not shown here). However, from Figs. 7 and 8 we see that it does not work as an indicator of success in general.

IR and IP seem to have a slight tendency to work better on the lower dimensional problems. They perform best on F_2. It is a bit disappointing that they, as well as wrapping, completely fail on F_5 in ten dimensions. They also have the interesting property of working well with very high σ_{init} values, most likely due to the downscaling of σ that is used in combination with them. Alternatively, it would also be possible to combine IP and IR with a penalty for the amount of constraint violation, as in [6]. However, that approach is very complicated and it is unclear if all this effort is necessary to obtain decent results.

Wrapping is the most successful repair function on F_{12}. On all problems, it achieves it's top performance with rather low σ_{init}. This is plausible, because with wrapping the maximal possible distance to the global optimum is only half of that with projection or reflection. So, especially if the problem is multimodal, (Baldwinian) wrapping may be beneficial through a higher exploratory power. However, the number of problems in this experiment is too small to verify this assumption. If the optimum lies on the edge of the feasible region, wrapping is problematic because it introduces cliffs. Langdon and Poli [5] show that this property causes problems to be difficult. Projection and reflection naturally avoid this difficulty by setting the decision variable to the respective value (projection) or creating an artificial basin that can guide the EA to the optimum from both sides (reflection).

3 Conclusions and Outlook

This work is the first systematic investigation on the interaction of repair functions for box constraints and inheritance models. The problem, dimension n, and step size σ_{init} also have a surprisingly strong influence, making final conclusions difficult. Further experiments would have to single out which problem characteristics are responsible for the observed effects. So far, it seems safe to draw the following conclusions regarding performance: If the global optimum lies on the edge or in a corner of the feasible space, projection would be the ideal choice. However, with its ability to complement the basin of attraction, reflection is a good alternative with acceptable performance throughout all problems. As the best approach will seldom be known beforehand and σ_{init} has a strong influence, it would be appealing to have an adaptive one that applies online learning.

In the future, a comparison to the more general constraint handling approaches [2] would be necessary to find out if repairing solutions is really worth the effort. Another interesting aspect would be the behavior under elitist selection, because there we should expect a much lower fraction of surviving repaired offspring. The homomorphous mapping between an arbitrary feasible region and a hypercube by Koziel and Michalewicz [14] could be used to transfer the proposed repair methods to a broader application area than box constraints. Also, note that it is always possible to convert a box constrained problem into an

unconstrained one by incorporating the repair function into the problem evaluation. This may be especially appealing to practitioners that need to get going quickly with an optimizer that is not well suited for constraints. Also, a specific theoretical treatment as by Lewis and Torczon [4] is unnecessary.

References

1. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
2. Arabas, J., Szczepankiewicz, A., Wroniak, T.: Experimental Comparison of Methods to Handle Boundary Constraints in Differential Evolution. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 411–420. Springer, Heidelberg (2010)
3. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore (May 2005), `http://www.ntu.edu.sg/home/EPNSugan`
4. Lewis, R.M., Torczon, V.: Pattern search algorithms for bound constrained minimization. SIAM Journal on Optimization 9(4), 1082–1099 (1999)
5. Langdon, W.B., Poli, R.: Evolving problems to learn about particle swarm optimizers and other search algorithms. IEEE Transactions on Evolutionary Computation 11(5), 561–578 (2007)
6. Hansen, N., Niederberger, A.S., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Transactions on Evolutionary Computation 13(1), 180–197 (2009)
7. Purchla, M., Malanowski, M., Terlecki, P., Arabas, J.: Experimental comparison of repair methods for box constraints. In: Proceedings of the 7th National Conference on Evolutionary Computation and Global Optimization. Warsaw University of Technology Publishing House (2004)
8. Schwefel, H.P.: Evolution and Optimum Seeking. Wiley, New York (1995)
9. Fletcher, R., Powell, M.J.D.: A rapidly convergent descent method for minimization. The Computer Journal 6(2), 163–168 (1963)
10. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)
11. Michalewicz, Z., Nazhiyath, G.: Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In: IEEE International Conference on Evolutionary Computation, pp. 647–651 (1995)
12. Beyer, H.-G., Sendhoff, B.: Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy –. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 123–132. Springer, Heidelberg (2008)
13. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA (March 2010)
14. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evolutionary Computation 7(1), 19–44 (1999)

Scalability of Population-Based Search Heuristics for Many-Objective Optimization

Ramprasad Joshi* and Bharat Deshpande

BITS, Pilani - K K Birla Goa Campus, Zuarinagar, Goa India 403726
{rsj,bmd}@goa.bits-pilani.ac.in

Abstract. Beginning with Talagrand [16]'s seminal work, isoperimetric inequalities have been used extensively in analysing randomized algorithms. We develop similar inequalities and apply them to analysing population-based randomized search heuristics for multiobjective optimization in $\mathbb{R}^n$ space. We demonstrate the utility of the framework in explaining an empirical observation so far not explained analytically: the curse of dimensionality, for many-objective problems. The framework makes use of the black-box model now popular in EC research.

Keywords: Multiobjective optimization, MOEA, Probability Measure Theory, Talagrand-type Inequalities.

1 Introduction

Evolutionary algorithms are complicated and show an extremely wide variety of behaviors. Multiobjective optimization by evolutionary algorithms (MOEAs) is an even more complex field of investigation. Hardness of problems for EAs has been the subject of intense and extensive research in more than a decade, and yet there has been no general framework for anaysis to which researchers adhere. Droste, Jansen, and Wegener [10] showed a very generalized framework for analysing randomized optimization heuristics which do not make use of any information about the function to be optimized. Their work [10,11], extended and enriched by Lehre and Witt [13] and Doerr and Winzen [8,9], has been successful in providing insight in the hardness of problems for randomized heuristics based on a black-box (oracular) use of the function to be optimized, or the efficiency of the latter. The aforementioned work is confined to single-objective optimization. The present work seeks to extend these ideas to the most general, multiobjective optimization, in continuous space. In order to analyse the performance of algorithms that essentially compute discrete sequences that are sampled from continuous random variables, relevant notions from measure theory are brought into use. The extended framework is demonstrated to be useful in explaining why most of the traditional mutation-crossover-based MOEAs fail miserably on many-objective problems when the number of objectives grows to 6 and beyond. To the authors' knowledge, this is the first attempt to apply Talagrand-type inequalities to population-based heuristic algorithms. For an introduction to probabilistic inequalities applied in analysis of algorithms, a recent book by Dubhashi and Panconesi [12] provides valuable and accessible material. For Talagrand's original work, see [16].

* The first author is partially supported by BITSAA, DST-GoI, and IIL, Pune, India.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 479–488, 2013.
© Springer-Verlag Berlin Heidelberg 2013

1.1 Preliminary Notation

First we describe some preliminary notation. We consider multiobjective optimization problems as defined below.

Definition 1. *A* **multiobjective optimization problem (MOP)** *is a black-box algorithm that, given an input $x \in \mathbb{R}^n$, decides whether it belongs to $\mathbb{X} \subsetneq \mathbb{R}^n$, a set constrained (decided) by a set of conditions $g_i(x) \geq 0, i = 1,2,\ldots,n_c$ and if it does, computes a mapping $f : \mathbb{R}^n \to \mathbb{R}^m$. The problem is to minimize f (more precisely, $f_i, i = 1,2,\ldots,m$) subject to conditions g.*

Here we are concerned only with the existence of a black-box algorithm (an oracle), so that, without regard to the cost of computing g or f, we can analyse the hardness of optimizing f. We call $\mathbb{X}$ the *search* or *design* space, and $f(\mathbb{X})$ the *objective* space. We assume that each of $\mathbb{X}$ and $f(\mathbb{X})$ are measurable and contained in compact sets (thus have finite Lebesgue measure). This is reasonable as practical engineering optimization problems as well as computable functions over reals (see [2]) will naturally satisfy these conditions.

Since there may not be any point $x* \in \mathbb{R}^n$ such that $f_i(x*) \leq f_i(x) \forall x \in \mathbb{X}, i = 1,2,\ldots,m$ simultaneously in general, the notion of Pareto-optimality is used in practice to compare various points in the search space. This notion is based on domination relationships. We define these notions before we turn to the class of algorithms to be considered in the next section.

Definition 2. $\forall x^{(1)}, x^{(2)} \in \mathbb{X}, x^{(1)} \preceq x^{(2)} \Leftrightarrow f_i(x^{(1)}) \leq f_i(x^{(2)}), i = 1,2,\ldots,m$ *We say* $x^{(1)}$ *weakly dominates* $x^{(2)}$.
$\forall x^{(1)}, x^{(2)} \in \mathbb{X}, x^{(1)} \preccurlyeq x^{(2)} \Leftrightarrow f_i(x^{(1)}) \leq f_i(x^{(2)}), i = 1,2,\ldots,m$; *and*
$\exists i \in \{1,\ldots,m\} : f_i(x^{(1)}) < f_i(x^{(2)})$ *We say* $x^{(1)}$ *dominates* $x^{(2)}$.
$\forall x^{(1)}, x^{(2)} \in \mathbb{X}, x^{(1)} \prec x^{(2)} \Leftrightarrow f_i(x^{(1)}) < f_i(x^{(2)}), i = 1,2,\ldots,m$ *We say* $x^{(1)}$ *strictly dominates* $x^{(2)}$.

Definition 3. *A set $O_f \subseteq \mathbb{X}$ such that $\forall x* \in O_f, \nexists x \in \mathbb{X}, x \preccurlyeq x*$ is called the* Pareto-optimal or nondominated set *of f in $\mathbb{X}$. $f(O_f)$ is its Pareto-optimal* front. *Both the notions can be restricted to subsets of $\mathbb{X}$ as well.*

2 Population-Based Randomized Search Heuristics for Multiobjective Optimization

Heuristic algorithms, especially randomized and bio-inspired algorithms, allow such a wide variation in their design and behaviors that seeking a general model is fraught with many potential limitations. Instead, here an attempt is made to model population-based heuristics that explicitly or implicitly make use of a probabilistic model of the function to be optimized built incrementally from samples of function values, obtained from a given auxiliary algorithm to compute them, at different points in the search space. Here a model black-box algorithm (Algorithm 1), adapted from Droste et al. [10], to represent the general case of multiobjective optimization in $\mathbb{R}^n$ space, is presented.

Algorithm 1 works in iterations called "generations" in evolutionary computation (EC) parlance. The set of search points I_t in each iteration t is called a "population".

Algorithm 1. Black-box $(\mu+\mu)$-Real MOEA

Require: $f : \mathbb{R}^n \supsetneq \mathbb{X} \to \mathbb{R}^m, m, n \geq 1$
Require: $f(\mathbb{X}) \subsetneq \mathbb{R}^m; \forall x \in \mathbb{X}, |f(x)| < \infty$
Require: $\mathscr{F}$, a σ-algebra on $\mathbb{X}$
Require: $P_0 : \mathscr{F} \to [0,1]$ an initial pdf
1: $t \leftarrow 0; X_0 \leftarrow Sample(P_0, \mathbb{X}, \mu); I_0 \leftarrow (X_0, f(X_0))$
2: **while** Stopping criteria are not met **do**
3: $P_{t+1} \leftarrow ComputeDistribution(P_t, I_t)$
4: $X_{t+1} \leftarrow Sample(P_{t+1}, \mathbb{X}, \mu)$
5: $I_{t+1} \leftarrow Update(I_t, (X_{t+1}, f(X_{t+1})), \mu)$
6: $t \leftarrow t+1$
7: **end while**
8: **return** I_t

The *ComputeDistribution* step in Line 3 can represent many pre-variation selection schemes, including, inter alia, parental selection like tournaments. The sampling and updation steps in lines 4 and 5 can (sometimes overlappingly) represent a fair variety of variation and survival selection schemes, including mutation, crossover, and Pareto-ranking-selection of the EC tradition. From the very large amount of literature available, refer to the comprehensive book by De Jong [5] for a broad outline. For the multitude of heuristics for multiobjective optimization, along with [3], two books [4] and [7] cover a lot of ground. At the end, note that Algorithm 1 directly models estimation of distribution algorithms (EDAs); for an introduction to EDAs, see [14,15]. We have not developed or used the concepts of black-box complexity or restrictions of Droste et al. [10,11], Lehre and Witt [13], and Doerr and Winzen [8,9] because our work is restricted to showing the applications of isoperimetric inequalities to the analysis of problems in the black-box model. We now seek to model the behaviors of algorithms on typical problems, i.e to model what De Jong [6] calls a "GAFO" or "Genetic Algorithm Function Optimization" situation.

3 Quantifying Exploration and Exploitation

In EC research, as well as in research on other randomized search heuristics, two themes, exploration and exploitation recur prominently. For comprehensive discussion on these themes, see Blum and Roli [1], and Deb [7, p.321]. Here we only note that in the beginning, a heuristic should try to expand the search in all directions in order to explore as much as possible, but after locating the nondominated set, it should confine search to the more promising areas, exploiting the knowledge gained during the earlier search. We develop these ideas in formal, precise ways in this section.

3.1 Search Spaces and Search Processes

The performance of Algorithm 1 on continuous functions can be analysed using a probabilistic model of the search process. For this, we map statements about domination in $\mathbb{R}^n$ and $\mathbb{R}^m$ space to bitstrings in $\{0,1\}^m$ space.

For any $x \in \mathbb{X}$, we define a family of functions $g_x : \mathbb{X} \to \{0,1\}^m$ as follows.

$$g_x(y)_i = \begin{cases} 0 & \text{if } f(y)_i < f(x)_i \\ 1 & \text{if } f(y)_i \geq f(x)_i \end{cases}, i = 1, \ldots, m$$

Next we define a function h to capture the notion that all but a limited few co-ordinates of a member z of $\{0,1\}^m$ are covered by subsets $A_1, A_2, \ldots, A_q \subseteq \{0,1\}^m$, for some $q > 1$, as follows:

$$h(A_1, \ldots, A_q, z) = \min \left[\left| \{ i : i \leq m, z_i \notin \{w_i^1, \ldots, w_i^q\} \} \right|, w^j \in A_j, j = 1, \ldots, q \right],$$

where the modulus $|.|$ stands for the cardinality of the set. The function h captures a certain sense of Hamming "distance" of z from the sets A_j. Since many variation operators in population-based randomized search heuristics explore neighbourhoods in terms of Hamming distance, this is relevant to analysing such heuristics.

Now we have the required machinery to bring into use one of the many concentration of measure bounds established by Talagrand [16]. Our main result is based on Talagrand [16]'s Theorem 3.1.1 that is discretized in the current context. Our proof is an adaptation of Talagrand [16, pp.113-115]'s proof. We use simpler ideas that become sufficient in the simplified discrete case. Note that our result is just a special case of Talagrand's result, whereas the original result is much more powerful and general. We close the section with a discussion of implications. In essence, the result gives us the likelihood of finding a point very much "different" in quality from some significant portion of the search space, telling us whether exploration is still due or exploitation can begin.

Theorem 1. *Let h be as defined above and $\mathbb{P}[.]$ be any probability distribution over $\{0,1\}^m$. Then*

$$\sum_{z \in \{0,1\}^m} q^{h(A_1, \ldots, A_q, z)} \mathbb{P}[\{z\}] \leq \frac{1}{\prod_{i=1}^q \mathbb{P}[A_i]}.$$

Proof. Induction on the dimension m.
Basis. For $m = 1$,

$$\sum_{z \in \{0,1\}} q^{h(A_1, \ldots, A_q, z)} \mathbb{P}[\{z\}] = q^{h_0} p_0 + q^{h_1} p_1,$$

where $p_0 = \mathbb{P}[\{0\}] = 1 - p_1 = 1 - \mathbb{P}[\{1\}]$, and h_z is shorthand for $h(A_1, \ldots, A_q, z)$. In the two-point space, A_i can be one of $\phi, \{0\}, \{1\}, \{0,1\}$, with probabilities $0, p_0, p_1, 1$ respectively. Straightforward computation shows that for any sequence of A_i, the result holds for any p_0.
Induction Hypothesis. Assume the result for $m = k \geq 1$:

$$\sum_{z \in \{0,1\}^k} q^{h(A_1, \ldots, A_q, z)} \mathbb{P}[\{z\}] \leq \frac{1}{\prod_{i=1}^q \mathbb{P}[A_i]}.$$

Step. For $m = k + 1$: Sets A_i are subsets of $\{0,1\}^{k+1}$. For $w \in \{0,1\}$ let

$$A_i(w) = \{(z_1, \ldots, z_k) : (z_1, \ldots, z_k, z_{k+1}) \in A_i\}$$

and projections

$$B_i = \{z \in \{0,1\}^k : \exists w \in \{0,1\},\ (z,w) \in A_i\}$$

of A_i on $\{0,1\}^k$. We denote $(z_1,\ldots,z_k,z_{k+1})$ by (z,w) where $w = z_{k+1}$.

Observe that $(z,w) \in A_i(w)$ can differ with the same co-ordinates of a member of A_i as of $A_i(w)$. Therefore, $h(A_1,\ldots,A_q,(z,w)) \leq h(A_1(w),\ldots,A_q(w),z)$, and by induction hypothesis,

$$\sum_{z\in\{0,1\}^k} q^{h(A_1,\ldots,A_q,(z,w))}\mathbb{P}[\{z\}] \leq \frac{1}{\prod_{i=1}^q \mathbb{P}[A_i(w)]}.$$

Similarly, $h(A_1,\ldots,A_q,(z,w)) \leq 1 + h(B_1,\ldots,B_q,z)$, and by induction hypothesis,

$$\sum_{z\in\{0,1\}^k} q^{h(A_1,\ldots,A_q,(z,w))}\mathbb{P}[\{z\}] \leq \frac{q}{\prod_{i=1}^q \mathbb{P}[B_i]}.$$

Thus, for $j = 1,\ldots,q$, if $C_{i\neq j,j} = B_i$ and $C_{j,j} = A_j(w)$ then

$$h(A_1,\ldots,A_q,(z,w)) \leq h(C_{1,j}\ldots,C_{q,j},z),$$

and hence by induction hypothesis, for $j = 1,\ldots,q$,

$$\sum_{z\in\{0,1\}^k} q^{h(A_1,\ldots,A_q,(z,w))}\mathbb{P}[\{z\}] \leq \frac{1}{\prod_{i=1}^q \mathbb{P}[C_{i,j}]}.$$

Summing the three inequalities over $\{0,1\}$ and combining, we get

$$\sum_{z\in\{0,1\}^{k+1}} q^{h(A_1,\ldots,A_q,(z,w))}\mathbb{P}[\{(z,w)\}] \leq$$

$$\sum_{w\in\{0,1\}} \left[\min_{j=1,\ldots,q}\left(\frac{1}{\prod_{i=1}^q \mathbb{P}[A_i(w)]}, \frac{q}{\prod_{i=1}^q \mathbb{P}[B_i]}, \frac{1}{\prod_{i=1}^q \mathbb{P}[C_{i,j}]}\right)\right] \mathbb{P}[\{w\}]$$

where the last term in the min stands for q terms for $j = 1,\ldots,q$. Looking at the indexed sets A,B,C, we can see that $|A_i| \leq 2|A_i(w)|$, $|A_i| \leq 2|B_i|$, and hence in $\{0,1\}^{k+1}$ space, too, $\mathbb{P}[A_i] \leq \mathbb{P}[A_i(w)]$ and $\mathbb{P}[A_i] \leq \mathbb{P}[B_i]$. Hence,

$$\min_{j=1,\ldots,q}\left(\frac{1}{\prod_{i=1}^q \mathbb{P}[A_i(w)]}, \frac{q}{\prod_{i=1}^q \mathbb{P}[B_i]}, \frac{1}{\prod_{i=1}^q \mathbb{P}[C_{i,j}]}\right) \leq \frac{1}{\prod_{i\leq q} \mathbb{P}[A_i]},\ \textit{thus},$$

$$\sum_{z\in\{0,1\}^{k+1}} q^{h(A_1,\ldots,A_q,(z,w))}\mathbb{P}[\{(z,w)\}] \leq \frac{1}{\prod_{i\leq q} \mathbb{P}[A_i]}.$$

Since w plays no role in $\mathbb{P}[A_i]$ the weighted (by probabilities) sum over $\{0,1\}$ on the RHS becomes the whole summand itself. That completes the induction. □

Corollary 1. $\forall k \geq 0$, $\mathbb{P}[\{z \in \{0,1\}^m : h(A_1,\ldots,A_q,z) \geq k\}] \leq \frac{1}{q^k \prod_{i\leq q} \mathbb{P}[A_i]}$.

Proof. By Theorem 1,

$$\mathbb{E}[q^{h(A_1,\ldots,A_q,z)}] \leq \frac{1}{\prod_{i=1}^{q} \mathbb{P}[A_i]}.$$

Now $\mathbb{P}[\{z : h(.,z) \geq k\}] = \mathbb{P}[\{z : q^{h(.,z)} \geq q^k\}] \leq \frac{\mathbb{E}[q^{h(.,z)}]}{q^k}$, the last inequality by Markov's inequality. □

Remark 1. – The bound developed above belongs to what are called isoperimetric inequalities. In abstraction (See [12, pp.126-127]), in a measure space Ω equipped with both a probability measure $\mathbb{P}[.]$ and a metric d, we have a t-fattening notion for a subset $A \subseteq \Omega$: $A_t := \{x \in \Omega : d(x,A) \leq t\}$. Then an abstract isoperimetric inequality asserts that

there is a "special" family of subsets $\mathscr{B}$ such that for any $A \subseteq \Omega$, $\forall B \in \mathscr{B}$,

$$\mathbb{P}[B] = \mathbb{P}[A] \Rightarrow \mathbb{P}[A_t] \leq \mathbb{P}[B_t].$$

- Here we are concerned with the "distance" (in terms of a quantitative measure of how many steps may be needed to reach them) of better search points from the regions that are explored so far or are worse than the current population.
- Since probabilities are fractional, the upscaling effect of the factor of $\prod_{i\leq q} \mathbb{P}[A_i]$ in the denominator of the bound obtained above must be gauged carefully for the bound to be of any significance. Simple computation shows that even if $q^{-\frac{k}{q}} \leq \mathbb{P}[A_i] \leq 2q^{-\frac{k}{q}}$, the bound is significant.
- Take $k = \frac{q}{2}$. For $h(.,z)$ to take values above k, the subsets A_i must have more than 2^k points each. Thus, for large m only the bound becomes more significant.
- For q disjoint subsets A_i, $\mathbb{P}[A_i] \leq \frac{1}{q}$. Thus $\frac{1}{q^k \prod_{i\leq q} \mathbb{P}[A_i]} \geq q^{q-k}$. Since $\mathbb{P}[h \geq m] = \mathbb{P}[h = m] \leq 1 - \sum_{i\leq q} \mathbb{P}[A_i]$, the bound is useless, for it is greater than 1 for q comparable to m. For disjoint sets, there are sharper bounds available via other techniques based on Chernoff-Hoeffding bounds and martingales, e.g. see [12].
- Let us take a concrete case. Let $m = 8$ and let all A_i be the same A, such that $\mathbb{P}[A] = \frac{1}{2}$ for $i \leq 32 = q$. What we are seeking here to estimate is the probability that a random point z differs from all populations of 32 points drawn from A in at least k co-ordinates. But the large size of the population makes the bound less useful: $\mathbb{P}[h \geq k] \leq 2^{32-5k}$ wherein the RHS will be above 1 unless $k \geq m-1$. Take $q = 8$ and we get $\mathbb{P}[h \geq k] \leq 2^{8-3k}$ and the bound is significant even for $k \geq 3$.

We discuss the applications of this result relevant to our task at hand subsequently.

3.2 Dividing up the Search Space

We need some more spare parts. Since f is considered to be continuous and bounded, probability measures Λ_X, Λ_f on $\mathbb{X}$ and $f(\mathbb{X})$ respectively are well-defined as restrictions of the suitably normalized Lebesgue measures on $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively. We also define two families of probability distributions $\mathbb{P}^x_X(.)$ and $\mathbb{P}^y_f(.)$ on $\{0,1\}^m$ induced by g_x as follows.

$$\forall A \subseteq \{0,1\}^m,\ \mathbb{P}^x_X(A) = \Lambda_X(\{y \in \mathbb{X} : g_x(y) \in A\})$$
$$\mathbb{P}^y_f(A) = \Lambda_f(\{z \in f(\mathbb{X}) : \forall x \in f^{-1}(y),\ g_x(z) \in A\})$$

It is obvious that $\mathbb{P}^x_X(A) = \mathbb{P}^y_f(A)$ whenever $y = f(x)$.

We first consider the probability that, given a point $x \in \mathbb{X}$ that dominates a significant portion $\mathscr{A}$ of $\mathbb{X}$, a point generated from x using some randomized variation operator dominates x. The significance of $\mathscr{A}$ is of course measured by $\Lambda_X(\mathscr{A}) \geq p$. Then, let $A = \{1^m\}$ be a singleton set. It represents $\mathscr{A} \cup \{x\}$. If we generate $y \in \mathbb{X}$ by the uniform distribution then the corresponding $z \in \{0,1\}^m$ is distributed by $\mathbb{P}^x_X(.)$. Thus $h_z = h(A,A,\ldots,A,z) = m$ represents the event that $y \preccurlyeq x$ for arbitrary repetitions of A. When A is repeated q times, $\mathbb{P}[\{h_z = m\}] \leq \frac{1}{q^m p^q}$. Minimizing over q, we get the tight bound at $q = -k/\log(p)$,

$$\mathbb{P}[\{z \in \{0,1\}^m : h_z = m\}] = \mathbb{P}[\{y \in \mathbb{X} : y \preccurlyeq x\}] \leq \left(-\frac{\log(p)}{mp^{\log(p)}}\right)^m$$

when x is such that $\Lambda_X(\{w \in \mathbb{X} : x \preccurlyeq w\}) = p$.

To understand the implications, let us again take a concrete case. For a point that dominates half the search space $\mathbb{X}$, the probability that a uniformly generated new point dominates it is bound above by $(1.8842/m)^m$. Thus the bound diminishes exponentially with the dimension m of the *objective* space.

The behavior of the bounds with change in p for each dimension m is shown in Figure 1.

Discussion. The bounds obtained in Theorem 1 are not useful unless we interpret them in the right context and seek probabilistic answers to practical questions of algorithm efficiency and problem hardness. First of all, let us take the simple case of a point $x \in \mathbb{X}$ dominating half of $\mathbb{X}$. We consider a purely randomized search algorithm and any

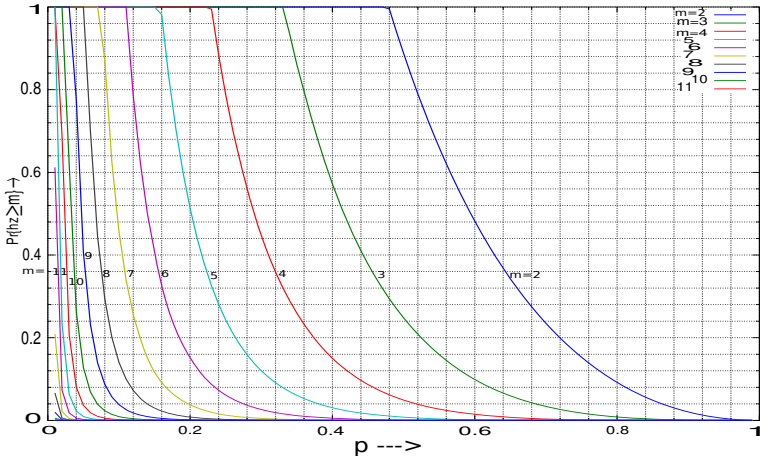


Fig. 1. The variation of the bound on $\mathbb{P}[\{z \in \{0,1\}^m : h_z = m\}]$ with p for various dimensions m

other, more intelligent algorithm. The first always keeps the distribution P_t in Algorithm 1 unchanged, at the uniform distribution, while the second one attempts to improve upon this by adapting P_t to the problem at hand based on the information obtained by all the search performed up to the given point. We call the first one RS and the alternative IRS. At low dimensions, 2 or 3, even when RS has sampled such a point, the ability to find without added intelligence a new better point is not severely restricted by our theorem but at even moderate dimensions of 5 and more, this ability is severely restricted. Whether really the probability is significant (i.e. the bound is a tight infimum) at low dimensions is not important here. The only conclusion we could draw from the bound at this point is that at dimensions 5 and more, purely randomized search will be grossly inefficient. But how much more efficient can be a more intelligent search? What conditions are necessary for that intelligence to work to more efficiency? The next section tries to answer these questions.

4 Bounding Algorithm Efficiency

Suppose it is desired that an IRS heuristic in Algorithm 1 works as follows. Beginning with a uniform random initial population, in each subsequent iteration $t > 0$ the heuristic computes the distribution P_t in Line 3 such that at a fixed minimum $\gamma\%$ of the times it does variation sampling (in Line 4), it samples a point that is better than all the points in I_t. That means, if $x_t^{(i)}$ represents a point in I_t, then

$$|\{1 \leq i \leq \mu : \forall 1 \leq j \leq \mu,\ x_{t+1}^{(i)} \preccurlyeq x_t^{(j)}\}| \geq \frac{\mu\gamma}{100}.$$

Now for I_t, let $y \in \mathbb{R}^m$ be such that

$$\forall j \leq \mu, \forall i \leq m, y_i \geq f_i(x_t^{(j)})$$

and

$$\forall i \leq m, \exists j \leq \mu, y_i = f_i(x_t^{(j)}).$$

Let us call y the *nadir* point of population I_t. Note that in general, it is *not necessary* that $y \in f(\mathbb{X})$. However, we can talk meaningfully (and measurably) of the portion of $f(\mathbb{X})$ that is strictly dominated by y; let its measure in the distribution P_{t+1} computed in Line 3 be p. Then $\forall i \leq \mu$

$$\Lambda_X(\{x \in \mathbb{X} : x_t^{(i)} \preccurlyeq x\}) \geq p.$$

Now we can bound the potential efficiency of a pure RS heuristic in Algorithm 1: the probability that a uniformly randomly generated point dominates all the μ points in I_t is

$$\mathbb{P}[\{h(A,A,\ldots,A,z) \geq 1\}] \leq \frac{1}{qp^q},$$

where $A = \{(1,1,\ldots,1)\}$. Minimizing over q, we get

$$\mathbb{P}[\{h(A,A,\ldots,A,z) \geq 1\}] \leq -p^{\frac{1}{\log p}} \log p.$$

The quantity on the RHS in this inequality actually represents an upper bound on the measure of points dominating a given current population, dependent only on the measure of points dominated by the current population. What happens when we consider points *strictly* dominating the current population as better, instead of just dominating points as better? Then we need to take the so-called *utopia* point u defined as:

$$\forall j \leq \mu, \forall i \leq m, u_i \leq f_i(x_t^{(j)})$$

and

$$\forall i \leq m, \exists j \leq \mu, u_i = f_i(x_t^{(j)}).$$

Of course, it is not necessary that $u \in f(\mathbb{X})$, and the part of $f(\mathbb{X})$ strictly dominated by u has measure say p' in P_{t+1}. Then, the bound obtained in the previous section is

$$\mathbb{P}[\{h(A,A,\ldots,A,z) = m\}] \leq \left(-\frac{\log(p')}{mp'^{\log(p')}}\right)^m. \tag{1}$$

Thus, for our IRS heuristic, presumably more intelligent than pure RS, to succeed in γ variation operations out of μ, if we define the notion of success as simple domination to strict domination, then the distribution P_{t+1} computed in Line 3 must be such that for any population I_t, then the measure of the space dominated by I_t must be $p \leq 0.6922^{\gamma}$. But we change the notion of success to strict domination, and then the variation success rate γ is bounded by the bounds given in the table below, obtained by maximizing over p' the bound in (1).

m	$\gamma \leq$
2	4.5949%
3	0.29184%
4	0.013196%
5	0.00046343%
6	0.000013308%
7	0.00000032321%
8	0.0000000068017%

This clearly shows the negative exponential dependence of the maximum success rate, that can be achieved by any intelligent heuristic on the general class of bounded measurable functions, on the objective dimension.

5 Conclusion

We have successfully shown the applications of isoperimetric inequalities of the Talagrand kind in the analysis of population-based randomized search heuristics for multi-objective optimization. We have explained the curse of dimensionality that cannot be avoided if the heuristic is ambitious enough to cover all bounded measurable functions. This work can be extended in many directions, especially to analyse the behavior of heuristics or to analyse hardness of problems restricted to narrower classes of functions, restricted by special isoperimetric inequalities. Such an analysis can even lead to refined heuristics for the restricted classes, especially of the EDA kind.

Acknowledgements. The authors are grateful to the anonymous reviewers and track chairs for helpful criticism. The first author acknowledges the help from BITS Alumni Affairs Division, from the Department of Science and Technology, Govt. of India, and from Innoventive Industries Ltd., Pune, India. The authors are grateful to Prof. Madhusudan V. Deshpande for his constant guidance.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35(3), 268–308 (2003)
2. Brattka, V., Hertling, P., Weihrauch, K.: A tutorial on computable analysis. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) New Computational Paradigms: Changing Conceptions of what is Computable. Springer (2008)
3. Coello Coello, C.A.: An updated survey of ga-based multiobjective optimization techniques. ACM Computing Surveys 32(2), 109–143 (2000)
4. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002) ISBN 0-3064-6762-3
5. De Jong, K.A.: Evolutionary Computaton: A Unified Approach. MIT Press, Cambridge (2006)
6. De Jong, K.A., Spears, W.M., Gordon, D.F.: Using Markov chains to analyze GAFOs. In: Whitley, L.D., Vose, M.D. (eds.) Proceedings of the Third Workshop on Foundations of Genetic Algorithms, pp. 115–137. Morgan Kaufmann, Estes Park (1994)
7. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley Student Edition. John Wiley and Sons Ltd., Singapore (2003)
8. Doerr, B., Winzen, C.: Memory-restricted black-box complexity. Electronic Colloquium on Computational Complexity 2011(Report No. 92) (2011)
9. Doerr, B., Winzen, C.: Towards a Complexity Theory of Randomized Search Heuristics: Ranking-Based Black-Box Complexity. In: Kulikov, A., Vereshchagin, N. (eds.) CSR 2011. LNCS, vol. 6651, pp. 15–28. Springer, Heidelberg (2011)
10. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Electronic Colloquium on Computational Complexity 2003 (Report No. 48) (2003)
11. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory of Computing Systems 39, 525–544 (2006)
12. Dubhashi, D.P., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms, Cambridge, NY (2009)
13. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Electronic Colloquium on Computational Complexity 2010 (Report No. 102) (2010)
14. Santana, R.: Estimation of distribution algorithms: from available implementations to potential developments. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2011, pp. 679–686. ACM, New York (2011), http://doi.acm.org/10.1145/2001858.2002067
15. Sastry, K., Goldberg, D., Pelikan, M.: Limits of scalability of multiobjective estimation of distribution algorithms. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2217–2224 (2005), doi:10.1109/CEC.2005.1554970
16. Talagrand, M.: Concentration of measure and isoperimetric inequalities in product spaces. Publ. Math. I.H.E.S. (81), 73–203 (1995)

On GPU Based Fitness Evaluation with Decoupled Training Partition Cardinality

Jazz Alyxzander Turner-Baggs and Malcolm I. Heywood

Faculty of Computer Science,
Dalhousie University, Halifax, Canada
{jazz,mheywood}@cs.dal.ca

Abstract. GPU acceleration of increasingly complex variants of evolutionary frameworks typically assume that all the training data used during evolution resides on the GPU. Such an assumption places limits on the style of application to which evolutionary computation can be applied. Conversely, several coevolutionary frameworks explicitly decouple fitness evaluation from the size of the training partition. Thus, a subset of training exemplars is coevolved with the population of evolved individuals. In this work we articulate the design decisions necessary to support Pareto archiving for Genetic Programming under a commodity GPU platform. Benchmarking of corresponding CPU and GPU implementations demonstrates that the GPU platform is still capable of providing a times ten reduction in computation time.

1 Introduction

The GPU approach to speed-ups for evolutionary computation (EC) as applied to supervised learning tasks such as classification or regression (function approximation) is approaching the point where increasingly sophisticated EC algorithms are being targeted e.g., [15,14,8,7]. However, one common underlying theme is the assumption that there is sufficient memory capacity on the target GPU device to support retention of all the training data. This tends to result in the wide-spread adoption of high end GPU cards (e.g., the nVidia Tesla series) that typically cost multiples of the host machine on which they reside ($\approx \$3,000$ per card). Thus, benchmarking can be performed on training data sets with millions of exemplars. However, at some point the training partition and / or the population of evolved individuals will need partitioning. Thus, swapping will be necessary between host CPU memory and GPU (a technique widely referred to as 'blocking' [9,11]); albeit with the caveat that such a transfer involve two different busses (host memory bus and PCI bus). Under such conditions, the speed-ups provided by the GPU are more difficult to achieve; at the very least the memory interface of multi-core CPUs has a lower handshaking overhead and greater throughput than that for the PCI bus. More generally, in order for GPUs to scale with data sets of arbitrary size – say, as in the case of data streams – an approach will need adopting that more directly embraces the need to *decouple* fitness evaluation from the size of the training partition.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 489–498, 2013.
© Springer-Verlag Berlin Heidelberg 2013

In this work we propose an approach based on Pareto archiving [6,4]. Specifically, Pareto archiving provides a mechanism for formulating a dual learning task, or identification of a subset of informative training exemplars that support the identification / development of a corresponding subset of learners. Such a process is explicitly coevolutionary, with both learners and subset of training instances (hereafter 'points') being represented as fixed size populations; thus Pareto archiving has the potential to decouple fitness evaluation from the cardinality of the training data.

The envisaged mode of operation is therefore for blocks of training data to be transferred to the GPU. Evolution will then take place relative to the current block content. The host makes decisions regarding size of blocks and retains the overall repository of training data. Periodically, the host will update the block content. In the meantime, the GPU concentrates on coevolving the point and learner populations. A breeder style of interaction will be assumed for controlling the turn over of point and learner content i.e., P_{gap} and H_{gap} points and GP individuals are replaced per generation. Thus, when block content changes, the point population will undergo a gradual change of content as more of the material from the new block is introduced and / or different point / GP individuals are identified as non-dominated.

A port of the above Pareto archiving methodology to will be undertaken using the CUDA computing platform and evaluated against the corresponding CPU implementation. Specifically an Nvidia GTX 660Ti graphics card($\approx$ \$350) is used. This Midrange card (based upon the Kepler architecture) is configured with 1355 cores at 1032 MHz and 2GB's of GDDR5 RAM. In particular, we target genetic programming for the learner representation and Symbolic Bid-based GP specifically (SBB) [13,5]. The SBB framework has been widely demonstrated under multiple data mining contexts, including large attribute spaces [5] and classification under streaming data constraints [1]. Moreover, SBB is already available in a code distribution with Pareto archiving directly supported and issues such as multi-class classification and task decomposition are addressed from the outset [13]. Thus, the essential question we attempt to answer is whether a GPU style acceleration of evaluation is still able to contribute a meaningful speed-up when both CPU and GPU implementations explicitly decouple fitness evaluation from training partition cardinality.

2 Overview to Symbiotic Bid Based GP

SBB assumes a three population model: points, teams and symbionts [13,5]. That is to say, the learners are represented by a team–symbiont partnership in which symbionts denote programs and teams declare 'teams' of programs. Thus the team population conducts a combinatorial search for good sets of programs whereas symbionts define the available population of programs. In addition, the specific form assumed for programs is that of bid based GP [12]. Thus, each symbiont consists of a linear GP program [2] and scalar action which, in this context, corresponds to a class label. Symbiont programs (hereinafter referred to

as 'programs') may only ever assume a *single* class label. A team must therefore consist of at least two programs with different class labels to avoid degenerate behaviour. Ideally, teams will learn to index a program with actions from each class. However, this is not formally enforced in order to let the team population gradually evolve the relevant content i.e., under multi-class contexts enforcing such a constraint might represent too complex a starting point (for experimental verification of this effect, see [1]). The variable length representation of teams therefore supports their incremental 'complexification'. A team is evaluated by executing all programs identified by the team and identifying the program with largest bid as the 'winner'. Such a program has won the right to suggest its action (class label) and evaluation proceeds with comparison against the known class label.

The point population is the subset of training exemplars over which fitness evaluation is performed. Each team is evaluated over all points at each generation. A fixed number of teams (H_{gap}) and points (P_{gap}) are added and removed at each generation as in a breeder approach to selection–replacement. Moreover, at each generation any program not receiving a team index is deleted, and new programs can be introduced by the search operator.

In the case of the team–point population interaction, the outcome of applying two teams to a common set of training exemplars (points) can be defined in terms of a pair of vectors, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Vector $\boldsymbol{v}_1$ is said to ***dominate*** vector $\boldsymbol{v}_2$ if the following relation holds,

$$dom(\boldsymbol{v}_1, \boldsymbol{v}_2) \Leftrightarrow \forall q : \boldsymbol{v}_1[q] \geq \boldsymbol{v}_2[q] \wedge \exists q : \boldsymbol{v}_1[q] > \boldsymbol{v}[q] \quad (1)$$

where q indexes members of the point population. Individuals with a dominance vector satisfying the above condition are said to be part of the Pareto front or the non-dominated set. Members of the front have the property that favouring one individual over another requires a tradeoff, implying that some training exemplars are more important than others. It is this property that facilitates the decoupling of the point population from the cardinality of the training partition as a whole. Pareto-coevolution therefore searches for candidate solutions that achieve high outcomes and points that serve as informative evaluators. For further details of Pareto archiving see [6,4] or for the specific case of the fitness sharing scheme used to maintain finite point and team populations in SBB see [13,5].

3 Architecture of gSBB

The GPU port for SBB – hereafter gSBB – assumes the basic control flow of the current SBB distribution[1] as summarized by Figure 1. Note that evolution and post training selection of a single 'champion' individual need not assume the same performance metric. Specifically, during 'EvaluatePrograms' the Pareto archiving methodology is assumed (Section 2), whereas any post training evaluation is performed relative to a single champion individual. For the purposes of

[1] `http://www.cs.dal.ca/~mheywood/Code/SBB`

this work we concentrate on the 'EvaluatePrograms' step as, without a speed-up here, any acceleration of the post training step remains a moot point.

Previously, the evaluation step entailed finding the highest bidding program for every point in the point population. If a new program was encountered in the team it was invoked against the point and the bid for that program–point pair recorded. In the end the action for the highest bidder was chosen and the system moved to (team) selection once every team was evaluated.

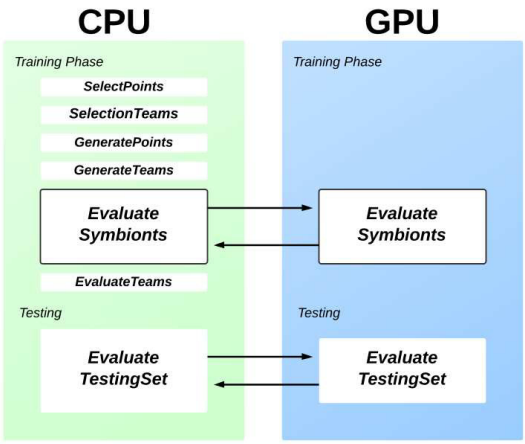


Fig. 1. SBB control flow and basic CPU–GPU partitioning of tasks

Under gSBB the process of program evaluation and team evaluation has been divided into two separate subtasks. That is to say, there the process of program execution can be separated from team membership. Thus, once the learners are evaluated the bids are returned to the CPU where the winning bids per team and corresponding action suggestion is resolved.

A significant source of deviation between gSBB and SBB is the data structuring used to store the populations. The SBB framework took an object oriented approach using nested classes to store data where it made most sense logically. That is a TeamObject contained a vector of program members (as well as its supplemental data such as ID and timestamp). While this made for clean code, it meant that data for each population was fragmented across memory, thus precluding efficient transfers to the GPU. To mitigate this overhead, the original SBB code was rewritten to use large allocated blocks of memory for each population so that all members of the point and team–program populations resided contiguously in memory. Not only does this reduce the overhead of mapping address space between the device and machine; it also allows for more advanced memory tuning such as page locked direct memory access (DMA), and coalescing data access without unneeded reformatting.

In the following we discuss specific GPU design issues associated with memory and thread management.

3.1 Problem Decomposition (Tiling)

Due to register usage and shared memory allocations, tasks are decomposed into 16×16 blocks of computation. This corresponds to 16 programs and 16 points, which produce a 2-d matrix of 256 bid values where each row is a single program applied over 16 points and each column is a point applied to each program. The size of these blocks has not been studied in depth but has been found to produce a high occupancy rate (usage of streaming multiprocessors) and matches the size of global memory reads. Global Memory accesses are performed in 64 byte reads. This reduces the number of reads to global memory, which is a costly operation.

3.2 Managing Global Memory Latency

To reduce the frequency of high latency global memory reads, the following approaches are assumed:

- As each instruction in the symbiont program is accessed multiple times during evaluation, learners are read into shared memory. This significantly reduces the number of global memory reads. Each read from global memory requires 400-600 cycles on the GPU, whereas 64 bits of shared memory is readable in a single clock cycle.
- On a load all 256 threads are used to preemptively load programs into shared memory given that each instruction is guaranteed to be accessed. While this could be performed for the point population as well, this becomes an expensive operation. Over-subscription of shared memory may reduce the overall parallelism, and given that all attributes of a point are not guaranteed to be accessed, this represents a reduction to the overall efficiency.
- To reduce the number of overall global memory reads, the point population was reorganized in memory based upon observed access patterns. Evaluating the instructions of a program requires a feature/attribute of the point to be read in from memory. As the programs are evaluated in parallel, multiple threads attempt to read from global memory at the same time. Each read from memory is of a fixed block size, and will satisfy as many requests possible for data in that block. As each thread requires the ith attribute of their corresponding point these reads are too far apart to be grouped together. This is analogous to being able to read a row of a matrix, but needing a column. To improve this the point matrix is re-organized such that the ith attributes are placed together for a block of points, effectively taking a transpose of a subset of points.

  The hardware being used performs 64 byte global memory reads, which means that it is possible to coalesce(group) sixteen 4-byte floating point numbers into a single memory read. This corresponds nicely as there are 16 points being evaluated in parallel for every program. To make this work, every group of 16 points is stored attribute-major order(attributes first), so that memory boundaries are aligned.

4 Results

A comparative study is performed between the Object Orientated implementation of SBB (hereafter cSBB) and the current development of gSBB(as summarized above.) In particular, this means that cSBB benefits from properties such as more efficient data structures for CPU as opposed to GPU memory management and, more significantly, cSBB caches the performance of team–point outcomes which do not change at each generation. At present, the gSBB exhaustively evaluates all programs on all points at each generation. This is potentially a significant advantage for cSBB as only P_{gap} and H_{gap} points and teams change per generation (where $P_{gap} < P_{size}/2; H_{gap} < H_{size}/2$). In short, in line with recent recommendations [10], SBB implementations associated with CPU and GPU follow from code bases optimized for their respective platforms.

Two data sets from the UCI repository will be employed: KDD99(Full) and Shuttle, with properties summarized by Table 3. The parameters for symbiont programs are outlined in Table 1 and the baseline configuration is outlined in 2. A total of four experimental variables will then considered: point population size, team population size, point population gap size, team population gapsize .

Table 1. Symbiont program configuration. These values are static and do not change during experimentation Parameters.

Parameter	Value
Maximum instructions per program	48
Register Count	8
Opcodes	$+, -, \times, \div, \log,$ $\mod, \exp, \cos$

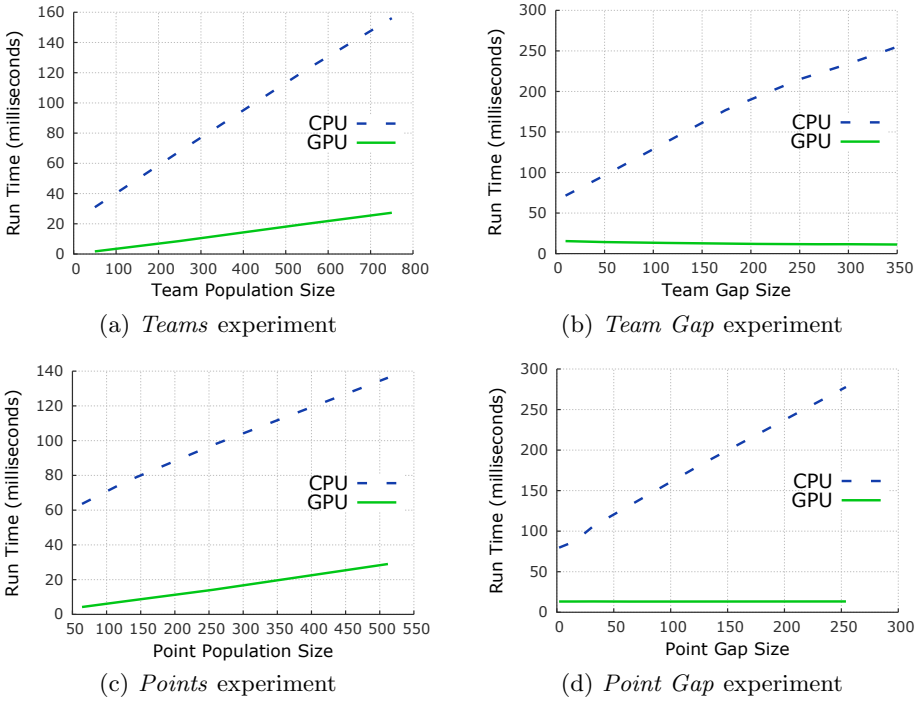
Table 2. Base Experiment Configuration. Gap sizes are set at a quarter the respective population size. Configuration.

Parameter	Value
Point Population Size (P_{size})	256
Point Gap Size (P_{gap})	64
Team Population Size (H_{size})	400
Team Gap Size (H_{gap})	100

The study focuses on runtime performance of each implementation. As both implementations follow from the same approach, there is no statistically significant difference in measured classification performance. Figures 2 and 3 summarize the performance of cSBB and gSBB in terms of average time *per generation* to complete training (milliseconds) for each data set. It can be seen in experiments in which only the population gap sizes were varied (subplots (b) and (d) respectively), no significant impact was observed in the case of gSBB. This is

Table 3. Benchmarking Data sets. Number of training and test exemplars; Number of attributes ($|A|$); Number of classes ($|C|$).

Dataset	#Train	#Test	$\|A\|$	$\|C\|$
KDD99	4 409 194	489 237	41	5
Shuttle	43 500	14 500	9	7

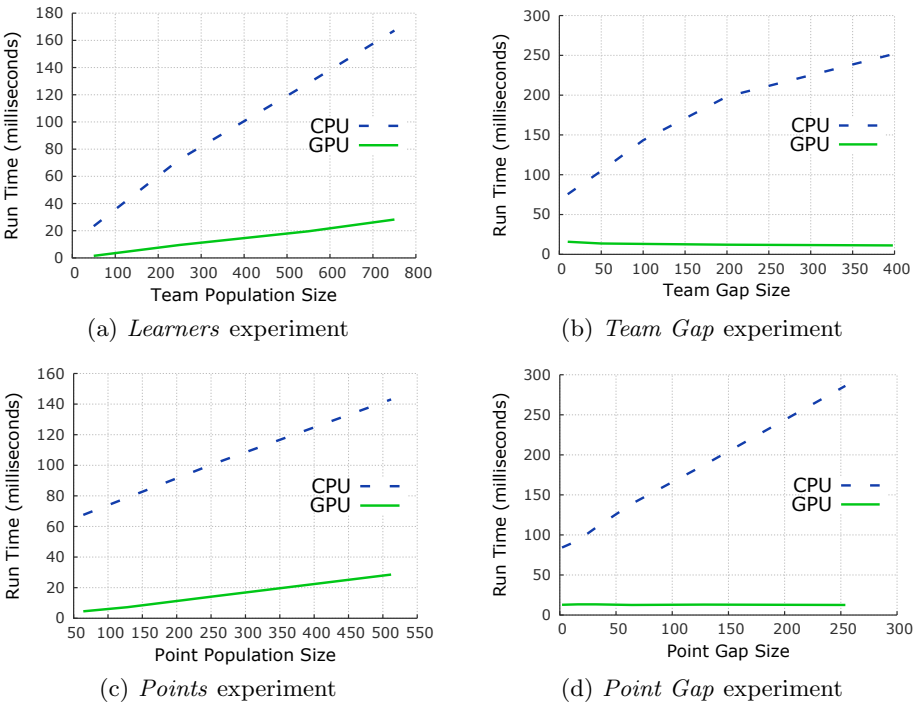


(a) *Teams* experiment (b) *Team Gap* experiment

(c) *Points* experiment (d) *Point Gap* experiment

Fig. 2. KDD99 data set

due the fact that gSBB does not cache outcomes from previous generations. To this effect the evaluation time remains constant as the workload is unchanged. The one caveat here is that an increase in team gap size causes a slight tapering of evaluation time. While the addition and deletion of programs from the population is a probabilistic event, which should maintain a linear correlation to the number of teams, the deterministic selection of the worst H_{gap} teams could cause a decrease in diversity in the program population. A higher team gap size (H_{gap}) forces teams to focus on successful programs early on in the evolutionary process, which in turn means more programs are being removed as they are no longer referenced. A smaller program population of course means less work needs to be performed and thus lowers evaluation time. Variation of all other parameters introduces an increased workload and subsequently causes a linear increase (albeit at different rates).



(a) *Learners* experiment (b) *Team Gap* experiment

(c) *Points* experiment (d) *Point Gap* experiment

Fig. 3. Shuttle data set

For example, increasing the number of individuals in the point population by 1 increases the total work by a factor proportional the the number of new programs per generation in the case of cSBB and by a factor proportional to the size of the program population in gSBB. While cSBB only needs to evaluate the new individuals in the population, it does see an increased evaluation time for increased population sizes even with a fixed sized population gap. This is due to the fact that SBB is a coevolutionary process and each new individual in one population must be evaluated against every individual in the other. This still manifests itself as a linear growth in evaluation time, however at a reduced rate.

In the cases where the number of features per point varies (i.e., through the use of different datasets), no significant change is observed. However, this is likely just a function of the relatively low attribute counts associated with KDD and Shuttle. Later research might include data sets with attribute counts measured in the thousands, in which case a significant impact would be expected [5]. As the control flow (or opcode selection) of the evaluation process is never dependant upon a computed value, the dataset has no direct effect on the amount of computation required. It does nevertheless require more data to be transferred between devices. Once again the use of Pareto archiving means the number of points is static and, in the case of Shuttle and KDD99 any overhead caused by

an increase in feature count is masked by asynchronous memory transfers and latency hiding. The caveat to this is that the size of the dataset has a significant impact upon the testing phase.

Considering all experiments performed, gSBB is an average $\times 10.83$ faster in the evaluation step compared to the CPU variant (cSBB). As a more useful metric, gSBB is capable on average of 7623.16 Program–Point pair evaluations per second.

5 Conclusion

Pareto archiving provides a mechanism for formally identifying a minimum subset of training exemplars for supporting a corresponding non-dominated set of GP individuals at each generation. Such a mechanism is potentially useful under a GPU context for continuing to scale GPU platforms to applications involving very large data sets or even an explicitly streaming data mining context. Specifically, the speed of evaluation associated with GPU platforms needs to be balanced against the overhead of PCI transfers. Current research either assumes that fitness evaluation on a CPU would be performed against the entire training partition or that the GPU has sufficient memory to retain the entire training partition. Conversely, coevolutionary mechanisms such as the host–parasite model (e.g., [3]) or Pareto archiving (as used here) demonstrate that fitness evaluation can be decoupled from the cardinality of the training partition. In this work benchmarking is performed to illustrate that meaningful speed-ups can still be attained when such a decoupling is employed on both GPU and CPU implementations of the SBB algorithm. This implies that commodity GPU cards can still represent an effective platform for accelerating GP fitness evaluation as more space will become available for retaining programs and data.

Future work will continue to develop the gSBB model. In particular having all of the SBB algorithm reside within the GPU and only require the CPU to periodically provide new blocks of data. In reaching this point gSBB would provide additional speedups, in particular with respect to the construction of the matrix of outcomes informing the definition for dominated versus non-dominated individuals. Specifically, the basic cost of performing the comparison central to Pareto archiving is n^2, as a matrix of outcomes needs first identifying (points) then all programs need comparing to identify the Pareto non-dominated teams and points. As the respective population size(s) increase the cost of evaluation also increases. However, the comparison operation itself is 'embarrassingly parallel', thus potentially open to GPU style speedups. Moreover, the direct memory cost of such an operation is still significantly lower than that of the training partition i.e., n is a function of learner–point population size not the training partition.

Acknowledgements. Authors gratefully acknowledge the support of the MITACS Internship and NSERC CRD programs.

References

1. Atwater, A., Heywood, M.I., Zincir-Heywood, N.A.: GP under streaming data constraints: A case for Pareto archiving? In: Proceedings of the ACM Genetic and Evolutionary Computation Conference, pp. 703–710 (2012)
2. Brameier, M., Banzhaf, W.: Linear Genetic Programming. Springer (2007)
3. Cartlidge, J., Bullock, S.: Combating coevolutionary disengagement by reducing parasite virulence. Evolutionary Computation 12(2), 159–192 (2004)
4. de Jong, E.D.: A monotonic archive for Pareto-coevolution. Evolutionary Computation 15(1), 61–93 (2007)
5. Doucette, J.A., McIntyre, A.R., Lichodzijewski, P., Heywood, M.I.: Symbiotic coevolutionary genetic programming: A benchmarking study under large attribute spaces. Genetic Programming and Evolvable Machines 13(1), 71–101 (2012)
6. Ficici, S.G., Melnik, O., Pollack, J.B.: A game-theoretic and dynamical-systems analysis of selection methods in coevolution. IEEE Transactions on Evolutionary Computation 9(6), 580–602 (2005)
7. Franco, M.A., Krasnogor, N., Bacardit, J.: Speeding up the evaluation of evolutionary learning systems using GPGPUs. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference, pp. 1039–1046 (2010)
8. Harding, S., Banzhaf, W.: Implementing Cartesian Genetic Programming Classifiers on graphics processing units using GPU.NET. In: ACM GECCO Computational Intelligence on Consumer Games and Graphics Hardware Workshop, pp. 463–470 (2011)
9. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A quantitative approach, 2nd edn. Morgan Kaufmann (1996)
10. Jaros, J., Pospichal, P.: A Fair Comparison of Modern CPUs and GPUs Running the Genetic Algorithm under the Knapsack Benchmark. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 426–435. Springer, Heidelberg (2012)
11. Langdon, W.B., Harrison, A.P.: GP on SPMD parallel graphics hardware for mega bioinformatics data mining. Soft Computing 12(12), 1169–1183 (2008)
12. Lichodzijewski, P., Heywood, M.I.: Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference, pp. 464–471 (2007)
13. Lichodzijewski, P., Heywood, M.I.: Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference, pp. 363–370 (2008)
14. Pospichal, P., Murphy, E., O'Neill, M., Schwarz, J., Jaros, J.: Acceleration of Grammatical Evolution using graphics processing units. In: ACM GECCO Computational Intelligence on Consumer Games and Graphics Hardware Workshop, pp. 431–438 (2011)
15. Shao, S., Liu, X., Zhou, M., Zhan, J., Liu, X., Chu, Y., Chen, H.: A gpu-based implementation of an enhanced GEP algorithm. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference, pp. 999–1006 (2012)

EvoSpace: A Distributed Evolutionary Platform Based on the Tuple Space Model

Mario García-Valdez[1], Leonardo Trujillo[1], Francisco Fernández de Vega[2], Juan J. Merelo Guervós[3], and Gustavo Olague[4]

[1] Instituto Tecnológico de Tijuana, Tijuana BC, Mexico
[2] Grupo de Evolución Artificial, Universidad de Extremadura, Mérida, Spain
[3] Universidad de Granada, Granada, Spain
[4] Centro de Investigación Científica y de Educación Superior de Ensenada, Ensenada BC, Mexico
{mario,leonardo.trujillo}@tectijuana.edu.mx,
fcofdez@unex.es, jmerelo@geneura.ugr.es, olague@cicese.mx

Abstract. This paper presents EvoSpace, a Cloud service for the development of distributed evolutionary algorithms. EvoSpace is based on the tuple space model, an associatively addressed memory space shared by several processes. Remote clients, called EvoWorkers, connect to EvoSpace and periodically take a subset of individuals from the global population, perform evolutionary operations on them, and return a set of new individuals. Several EvoWorkers carry out the evolutionary search in parallel and asynchronously, interacting with each other through the central repository. EvoSpace is designed to be domain independent and flexible, in the sense that in can be used with different types of evolutionary algorithms and applications. In this paper, a genetic algorithm is tested on the EvoSpace platform using a well-known benchmark problem, achieving promising results compared to a standard evolutionary system.

Keywords: Distributed Evolutionary Algorithms, Tuple Space, Cloud Computing.

1 Introduction

Currently, computational resources are offered as different types of services within the Cloud computing model (CCM) [1]. For example, infrastructure as a service (IaaS) such as Amazon's EC2, or Platform as a Service (PaaS) such as Google's App Engine. However, within Evolutionary Computation (EC), most algorithms have not been ported to the CCM. The standard are local, serial and synchronized algorithms, instead of distributed, parallel and asynchronous systems. Therefore, new proposals that exploit the CCM would be an important contribution within the field, since it lends itself to models of evolution that can encourage population dynamics not present in standard algorithms.

This paper presents EvoSpace, a population store for the development of evolutionary algorithms (EA) that are intended to run on the Cloud. Within the

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 499–508, 2013.
© Springer-Verlag Berlin Heidelberg 2013

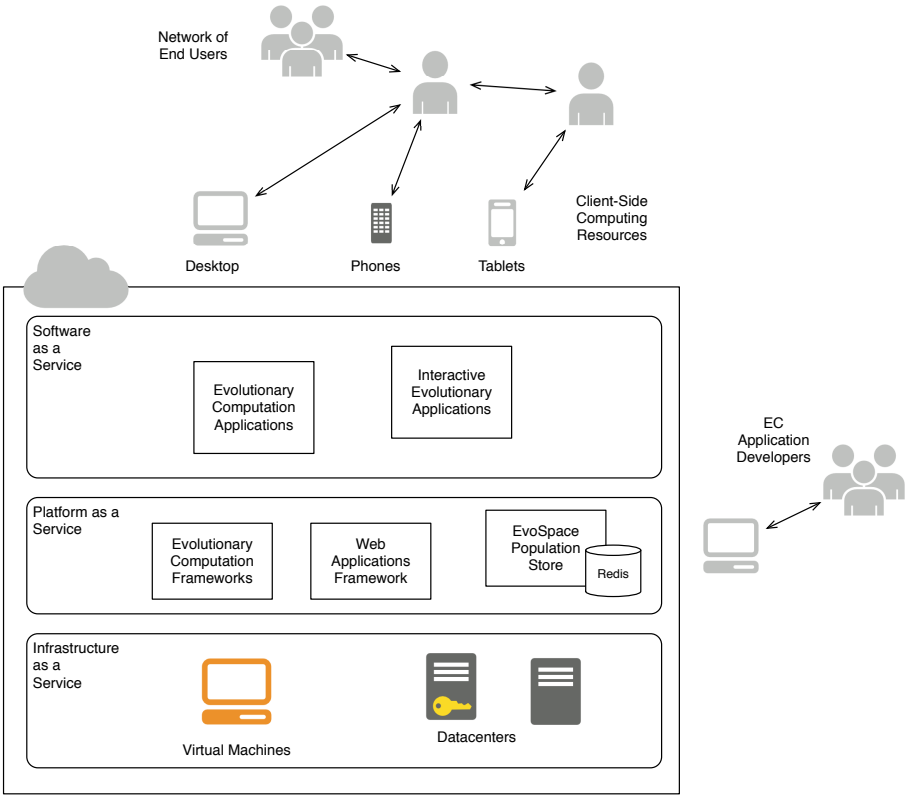


Fig. 1. Conceptual diagram of the CCM and of how EvoSpace can fit within, using it as a PaaS or possible using it to develop a SaaS

CCM, EvoSpace is conceived as a Platform as a Service (PaaS) component, where users can create populations of individuals on demand without the need to install software or invest in additional hardware. EvoSpace is designed to be versatile, since the system that manages the population is decoupled from any particular EA. Client processes, called EvoWorkers, dynamically and asynchronously interact with the EvoSpace store and perform the required evolutionary routines. EvoWorkers can reside on remote clients or on the platform server. Software as a Service (SaaS) applications could also be developed using EvoSpace, where users could run entire experiments on the Cloud or implement interactive applications that are completely hosted by the service (see [5]). EvoSpace is well suited for this kind of environment, since it can be accessed by any web enabled device and is robust to lost connections with remote clients. Figure 1 presents a conceptual diagram of how EvoSpace fits within the CCM.

This work presents a proof of concept implementation of EvoSpace. EvoSpace is tested on standard benchmark problems for genetic algorithms, analyzing performance, efficiency and diversity. The remainder of the paper proceeds as

follows. Section 2 reviews related work. Afterwards, Section 3 describes the EvoSpace platform and implementation details. The experimental work is presented in Section 4. Finally, conclusions are given in Section 5.

2 Related Work

The present work builds on several proposals of distributed and interactive EAs. Langdon [8] proposed a distributed EA of artificial agents expressed as fractal structures. That work employs a global population residing on a central web server that distributes portions of the population using Javascript. Similar pool-based approaches can be traced back to the A-Teams system [15], which is not restricted to EAs. Another proposal is made by G. Roy et al. [13], who developed a multi-threaded system with a shared memory architecture that is executed within a distributed environment. On the other hand, Bollini and Piastra [2] emphasize persistence over performance, proposing a system that decouples population storage from the evolutionary operations. A similar model is proposed by Merelo et al. [11], who use a web-server to access a population stored on a database. Klein and Spector [7], and Merelo at al. [11], also propose a Javascript implementations for EAs. The goal of these papers is to distribute fitness evaluations over the web. More recently, Cotillon et al. [4] extended browser-based EAs to the Android OS. The work by Langdon [8] is also an interactive EA, where the goal is to evolve virtual creatures. Similarly, Secretan et al. [14] and Clune and Lipson [3], use web-based interactive EAs to evolve artistic artifacts. In both cases, user (connected client) collaboration is encouraged. The most recent comparable work with EvoSpace is the SofEA system of Merelo et al. [9,10]. SofEA is an EA mapped to a central CouchDB object store. It provides an asynchronous and distributed search process, where the evolutionary operators are decoupled from the population. An important difference is that in SofEA all the information regarding the evolutionary progress is continuously updated on the central repository, while in EvoSpace the population container does not store knowledge regarding the search.

3 EvoSpace

EvoSpace consists of two main components. First, the EvoSpace container that stores the evolving population. The second component consists of the remote clients called EvoWorkers, which execute the actual evolutionary process, while EvoSpace is only a population repository. In a basic configuration, EvoWorkers extract a small subset of the population, and use it as the initial population for a local EA executed on the client machine. Afterwards, the evolved population from each EvoWorker is returned to the EvoSpace container. The server-side ReInsertionManager process is used to alleviate possible problems that might occur during the search; for instance, when the EvoSpace container starves or connections are lost. This can be done because a copy of each sample is stored in a priority queue used by ReInsertionManager to re-insert the sample to the central

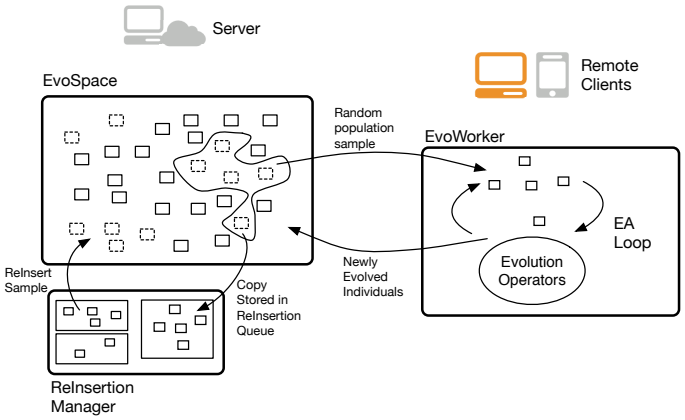


Fig. 2. Main components and dataflow within EvoSpace

population; similar to games where characters are respawned after a certain time. Figure 2 illustrates the main components and dataflow within EvoSpace.

The EvoSpace container. EvoSpace is based on the tuple space model, an associatively addressed memory space shared by several processes. A tuple space can be described as a distributed shared memory (DSM) abstraction, organized as a *bag* of tuples. A tuple t is the basic tuple space element, composed by one or more fields and corresponding values. In this model, the basic operations that a process can perform is to insert or withdraw tuples from the tuple space.

EvoSpace is composed by a set of objects ES and a set of interface methods provided by a central server. For an EA system, objects correspond to individuals and all their corresponding features and related information. Objects can be withdrawn, processed and replaced into ES using a specified set of methods. However, EvoSpace is different from other tuple space implementations in the sense that retrieving and reading objects from ES are random operations. Individual objects are not of high interest when accessing ES. Therefore, EvoSpace offers the following interface methods.

Read(n): This method returns a random set A of objects from ES, with $|A| = n$ and $A \subset ES$, if $n < |ES|$, the method returns ES otherwise.

Take(n): Returns a random set A, following the similar logic used for $Read()$. However, in this case the sequence of $Take()$ operations provide a temporal dimension to the dynamics of set ES. We can define ES_i as the set at the moment of the $i-th$ $Take()$ operation and A_i as the output. The contents of EvoSapce are then given by $ES_{i+1} = ES_i \setminus A_i$; i.e., the objets taken are effectively removed from ES. The objects taken are also copied to a new set S_i of *sampled objects* and stored within a temporary collection $\mathcal{S}$ on the server, implemented as a priority queue. Sets $S_i \in \mathcal{S}$ can then be reinserted to ES if necessary.

ReInsert(i): This method is used to reinsert the subset of elements removed by the $i-th\ Take()$ operation, such that the contents of EvoSpace are now $ES \cup S_i$ if $S_i \in \mathcal{S}$ and ES is left unchanged otherwise.

Insert(A): This method represents the union operation $ES \cup A$.

Replace(A,i): Similar to $Add()$, however set A should be understood as a replacement for some $S_i \in \mathcal{S}$, hence $|A| = |S_i|$, but the objects in A can be different (evolved) objects from those in S_i. Moreover, if S_i exists it is removed from $\mathcal{S}$. However, if S_i does not exist this means that a $ReInsert(i)$ operation preceded it, this increases the size of ES.

Remove(A): This method removes all of the objects in A that are also in ES, in such a way that the contents of EvoSpace are now set to $ES \cup (A \cap ES)$.

Individuals. As stated above, objects in ES represent individuals in an EA. Explicitly, the objects in ES are stored as *dictionaries*, an abstract data type that represents a collection of unique keys and values with a one to one association. In this case, keys represent specific properties of each object and the values can be of different types, such as numbers, strings, lists, tuples or other dictionaries. In the current implementation, the objects in ES are described by the following basic fields. An **id** string that represents a unique identifier for each object. A **chromosome** string, which depends on the EA and the representation used. The **fitness** dictionary for each individual; allowing the system to store multiple fitness values for multi-objective or dynamic scenarios. A **parents** dictionary with identifiers of the individual(s) from which it was produced. Finally, a **GeneticOperator** string that specifies the operator that produced it.

The EvoSpace Server Processes. On the server side, a process called **EvoSpaceServer** is executed, which creates and activates a new EvoSpace container object and waits for requests to execute interface methods. Additionally, on the server three more processes are executed, these are: **InitializePopulation**, **ReInsertionManager** and **EvolutionManager**. **InitializePopulation** is executed once, its goal is obvious, initialize the population by adding *popsize* random individuals. The function that creates new individuals depends on the problem and representation used. **ReInsertionManager** is used as a failsafe process that periodically checks (every wt seconds) if the size of the population in ES falls below a certain threshold or if the time after the last $ReInsert()$ is greater than $next_r$. If any of these conditions are satisfied, then rn subsets $S_i \in \mathcal{S}$ are reinserted into ES using the $ReInsert()$ method. Moreover, the **ReInsertionManager** makes a $ReInsert(i)$ call when $EvoWorker_i$ (see below) looses a connection with the server, thus recovering the population sample. Finally, **EvolutionManager** periodically checks if a termination condition is satisfied, which is checked by the $isOver()$ method. This method can be implemented according to the needs of the evolutionary search. For instance, $isOver()$ can check if an optimal solution is found or if a maximum number of function evaluations have been performed.

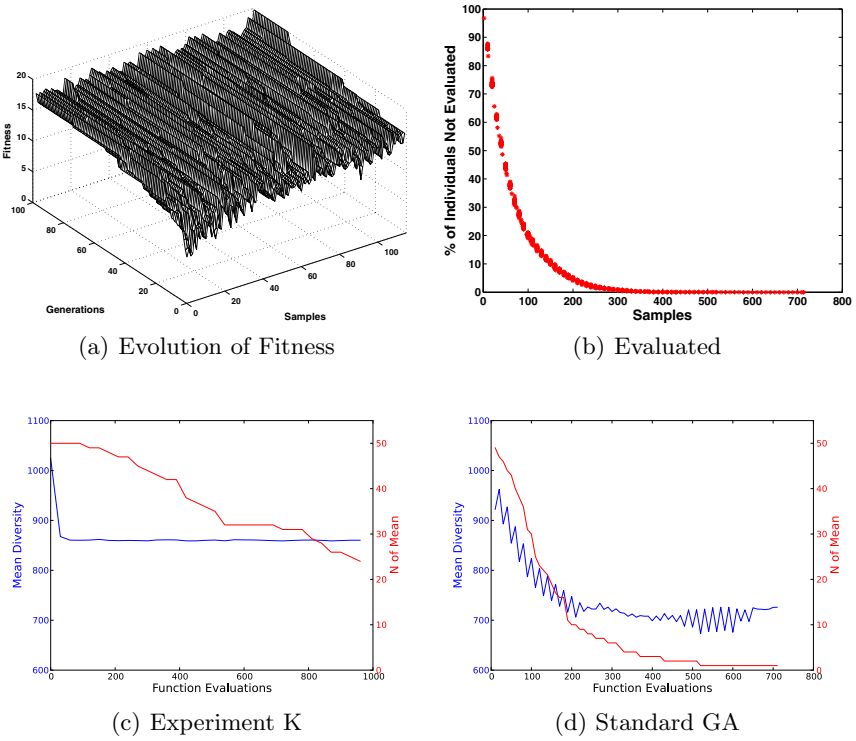
The EvoSpace Clients: EvoWorkers. The second component of the proposed model are the processes executed on client machines, referred to as **EvoWorkers**. Each client loads a process that executes the main code of the EA. The **EvoWorker** process is straightforward, it requests a set of objects A_i from the ES container. Afterwards, the $Evolve()$ function is called where the actual evolutionary cycle is performed. In this scenario, A_i can be seen as a local population on which evolution is carried out for g generations. The result of this evolution is then returned and reinserted into ES, then the EvoWorker can request a new set from ES and repeat the process. Otherwise, each EvoWorker could specialize on a particular part of the evolutionary process, such as selection, evaluation or genetic variation; an approach not taken in the present work.

Implementation. Individuals are stored in-memory, using the Redis key-value database. Redis was chosen over a SQL-based management system, or other non-SQL alternatives, because it provides a hash based implementation of sets and queues which are natural data structures for the EvoSpace model. For example, selecting a random key from a set has a complexity of O(1). The logic of EvoSpace is implemented as a python module exposed as a Web Service using cherrypy and django http frameworks. The EvoSpace web service can interact with any language supporting json-rpc or ajax requests. The EvoSpace modules and workers in JavaScript, JQuery and python are available with a Simplified BSD License from https://github.com/evoWeb/EvoSpace.

4 Experiments and Results

The goal of the experimental setup is to analyze the performance of EvoSpace on a binary GA. To do so, we use a common benchmark problem from evolutionary computation literature, the K-trap function [16]. This problem presents multiple local-maxima and a deceptive fitness landscape. Table 1 summarizes the different experimental configurations tested in this work, based on the K value, number of EvoWorkers, the sample size taken by each worker and the chromosome length used in each case. The number of individuals in the EvoSpace repository is set to 1024 for 4-trap experiments and 4096 for 5-trap. Moreover, Table 2 summarizes the shared parameters and algorithmic configuration; with common values for a GA search. The maximum number of samples that each EvoWorker can take from EvoSpace is 1000. For comparison, a standard GA is also used for each benchmark problem. For the 4-trap problem, the maximum number of generations is 4000, for the 5-trap problem it is 1000, this bounds the number of function evaluations based on the maximum number of samples taken from all of the EvoSpace runs. The remaining GA parameters are set with the values of Table 2. In total, 50 runs were performed of each experimental configuration.

First, to visualize how fitness evolves over all of the samples taken from EvoSpace see Figure 3a. This figure shows the evolution of best-fitness for a single run of experiment K; the analysis focuses on a single run instead of the mean of all runs in order to emphasize the local dynamics of the evolutionary



(a) Evolution of Fitness (b) Evaluated

(c) Experiment K (d) Standard GA

Fig. 3. (a) Evolution of fitness for a run of experiment K; the plot shows how fitness evolves for each sample taken by the EvoWorkers. (b) Scatter plot, considering all runs, of the percentage of non-evaluated individuals from experiment K. (c) Evolution of diversity for the basic GA on the 5-trap problem. (d) Evolution of diversity in EvoSpace for experiment K. Plots c-d are double y-axis plots, showing the mean diversity over all runs (dark line) and the number of runs N used to compute the mean.

process. The plot shows how fitness evolved on each EvoWorker that participated in the search. Evolution of fitness is organized based on the two temporal axis of the horizontal plane. With respect to the sample number, independent of which EvoWorker took the sample, and with respect to the generations of the evolutionary process executed on the corresponding EvoWorker. In other words, these plots provide a collective view of the evolutionary process from the perspective of all EvoWorkers. Since the global optimum is a fitness value of 10, we can see that the evolution on the last sample taken from EvoSpace reaches the global optimum. The overall performance of EvoSpace is summarized in Table 3, which shows the total number of runs, out of 50, that found the global optima. EvoSpace outperforms the standard GA on both functions, with a substantial increase in the number of optima found.

Besides the performance with regards to fitness, some other questions are pertinent to ask of the entire EvoSpace population. In this case, different measures can

Table 1. Different experimental configurations used to test the performance of EvoSpace

Experiment	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
K-trap	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5
EvoWorkers	1	1	4	4	8	8	16	16	32	1	4	8	16	32	40
Sample size	32	64	32	64	32	64	32	64	32	64	64	64	64	64	64
Chromosome length	40	40	40	40	40	40	40	40	40	50	50	50	50	50	50

Table 2. Parameters and algorithm configurations for all experiments

Parameter	Maximum Function Evaluations	Crossover (Prob.)	Mutation (Prob.)	Generations per EvoWorker
Value	100 Gens per worker	Single point (1)	Point (0.06)	100

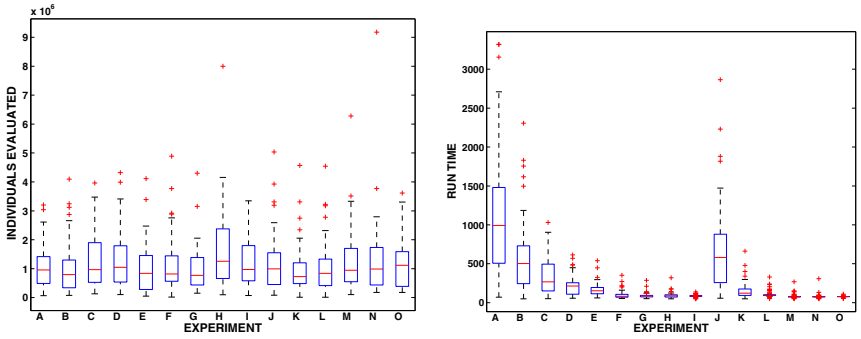
Table 3. Different experimental configurations used to test the performance of EvoSpace. GA-K are the baseline GA results for the 4-trap and 5-trap functions respectively.

Experiment	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	GA-4	GA-5
Optima found	48	50	50	49	49	50	48	50	50	50	50	50	50	50	50	34	29

be tracked during evolution, however the progression of each performance measure must be taken with respect to the number of samples taken from EvoSpace or the number of fitness function calls. For instance, since every EvoWorker takes a random sample of individuals, one concern might be that some individuals in the initial population might not be chosen and evaluated, and valuable genetic material wasted. However, as shown in Figure 3b, the percentage of individuals within EvoSpace that have not been evaluated quickly decreases with respect to the number of samples. Another interesting question regards diversity, how does it evolve within EvoSpace. Here, diversity is given by the sum of the pairwise Hamming distances of all individuals within the population [12]. Figure 3c shows how diversity evolves for experiment K. For comparison, Figure 3d presents the evolution of diversity for the basic GA[1] These plots show how the standard GA has a problem converging towards the optimum, progress seems to halt after the initial generations; this explains the poor performance achieved on this problem. On the other hand, EvoSpace continuously converges towards the global optimum, as evidenced by the high number of runs that ended with the global optimum.

A comparison of the amount of computational effort required in each experiment is given in Figure 4, which shows boxplots of all runs in each experiment. Figure 4a plots the total number of individuals evaluated in each run, which is similar in all experiments. Finally, Figure 4b compares the total run time shown

[1] The time scale is given by the number of evaluated individuals, in increments that correspond to the number of individuals evaluated in 10 samples from experiment I.



(a) Evaluated Individuals (b) Run-time

Fig. 4. (a) Number of evaluated individuals. (b) Total run-time.

in seconds. Run-time is reduced significantly by the number of EvoWorkers that participate in the search, as expected.

5 Conclusions

EvoSpace proposes a PaaS system for Cloud-based EAs, where the search is carried out as a distributed and asynchronous process. EvoSpace is tested on a standard GA benchmark. While the search conducted by an EvoSpace-based EA is fundamentally different from a standard sequential EA, the search is able to produce competitive results with regards to solution quality. Moreover, recent work has also shown that EvoSpace can acomodarte a less common type of EA, an interactive system for artistic design [5]. While initial results are encouraging, future work must evaluate fault tolerance and the effects of different algorithm parameters and settings. Also, other features will be incorporated to Evospace, such as multiple populations management, authentication, quotas and throttling. Finally, to contextualize the main strengths and weaknesses of EvoSpace a comprehensive empirical evaluation must be performed with other benchmark tests, compared with other distributed and web-based EAs.

Acknowledgments. Research supported by DEGEST-ProIFOPEP (Mexico) Research Project 4616.12-P; CONACYT (Mexico) Basic Science Research Project No. 178323; Regional Government Junta de Extremadura, Consejería de Economía-Comercio e Innovación and FEDER, project GRU09105; projects TIN2011-28627-C04-02 and -03 (ANYSELF), awarded by the Spanish Ministry of Science and Innovation; and project P08-TIC-03903 awarded by the Andalusian Regional Government.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)

2. Bollini, A., Piastra, M.: Distributed and persistent evolutionary algorithms: A design pattern. In: Proceedings of the Second European Workshop on Genetic Programming, pp. 173–183. Springer, London (1999)
3. Clune, J., Lipson, H.: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In: Proceedings of the European Conference on Artificial Life, pp. 144–148 (2011)
4. Cotillon, A., Valencia, P., Jurdak, R.: Android Genetic Programming Framework. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds.) EuroGP 2012. LNCS, vol. 7244, pp. 13–24. Springer, Heidelberg (2012)
5. Garcia, M., Trujillo, L., Fernández-de-Vega, F., Merelo-Guervós, J.J., Olague, G.: EvoSpace-Interactive: A Framework to Develop Distributed Collaborative-Interactive Evolutionary Algorithms for Artistic Design. In. In: Proceedings of the 2nd International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design, EvoMUSART (2013)
6. Gelernter, D.: Generative communication in linda. ACM Trans. Program. Lang. Syst. 7(1), 80–112 (1985)
7. Klein, J., Spector, L.: Unwitting distributed genetic programming via asynchronous javascript and xml. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 1628–1635. ACM, New York (2007)
8. Langdon, W.B.: Global Distributed Evolution of L-Systems Fractals. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 349–358. Springer, Heidelberg (2004)
9. Merelo-Guervós, J.-J., Mora, A., Cruz, J.A., Esparcia, A.I.: Pool-Based Distributed Evolutionary Algorithms Using an Object Database. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 446–455. Springer, Heidelberg (2012)
10. Merelo-Guervos, J.J., Mora, A., Cruz, J.A., Esparcia-Alcazar, A.I., Cotta, C.: Scaling in distributed evolutionary algorithms with persistent population. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE Computer Society (June 2012)
11. Merelo Guervos, J.J., Valdivieso, P.A.C., Laredo, J.L.J., Garca, A.M., Prieto, A.: Asynchronous distributed genetic algorithms with javascript and json. In: IEEE Congress on Evolutionary Computation, pp. 1372–1379. IEEE (2008)
12. Morrison, R.W., De Jong, K.A.: Measurement of Population Diversity. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 31–41. Springer, Heidelberg (2002)
13. Roy, G., Lee, H., Welch, J.L., Zhao, Y., Pandey, V., Thurston, D.: A distributed pool architecture for genetic algorithms. In: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation, CEC 2009, pp. 1177–1184. IEEE Press, Piscataway (2009)
14. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: A case study in collaborative evolutionary exploration of design space. Evol. Comput. 19(3), 373–403 (2011)
15. Talukdar, S., Baerentzen, L., Gove, A., De Souza, P.: Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics 4(4), 295–321 (1998)
16. Thierens, D.: Scalability problems of simple genetic algorithms. Evolutionary Computation 7, 331–352 (1999)

Cloud Driven Design of a Distributed Genetic Programming Platform

Owen Derby, Kalyan Veeramachaneni, and Una May O'Reilly

Massachusetts Institute of Technology, USA.
{ocderby,kalyan,unamay}@csail.mit.edu

Abstract. We describe how we design FlexGP, a distributed genetic programming (GP) system to efficiently run on the cloud. The system has a decentralized, fault-tolerant, cascading startup where nodes start to compute while more nodes are launched. It has a peer-to-peer neighbor discovery protocol which constructs a robust communication network across the nodes. Concurrent with neighbor discovery, each node launches a GP run differing in parameterization and training data from its neighbors. This factoring of parameters across learners produces many diverse models for use in ensemble learning.

Keywords: cloud computing, machine learning, genetic programming, distributed evolutionary computation.

1 Introduction

Recent availability of on-demand massive compute resources, i.e. *clouds*, has encouraged research into massive parallelization of machine learning (ML) systems (see *Mahout* and *GraphLab* [7,8]). Because clouds are different from other distributed resources like clusters or grids, they impose new requirements on how we parallelize ML, but also offer new opportunities[1]. In a cloud, the time to acquire many nodes is often quite long. To efficiently use a cloud, ML needs to start as soon as the first instance is acquired and not wait for the final instance's acquisition. When learning for an extended period of time, it needs to be designed as an online system where best interim results can be obtained at any time. It needs to expand or contract by taking advantage of cloud's elasticity to respond to varying resource costs and handle node failures.

These requirements drive innovation in both cloud-scale evolutionary computation (EC) and any machine learning enabled by EC. In this contribution we present FlexGP which is an elastic, cloud-scale, distributed platform using Genetic Programming for ML (GPML). FlexGP is a new version of previous work in [9]; we will refer to the old version as FlexGP-ECJ to distinguish between the two. We focus on GPML here, but any EA could be used with FlexGP. GPML is not well-suited for use with any of the existing cloud parallelization frameworks. Some of these frameworks, like MapReduce, were designed for completing large, single-run computations and do not readily

[1] These include the ability to learn many heterogeneous models very cheaply and the ability to learn for an extended period of time.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 509–518, 2013.
© Springer-Verlag Berlin Heidelberg 2013

accommodate the iterative nature of GPML. Others were retrofitted from grid or cluster architectures and thus do not fully realize the potential of the cloud. An example is FlexGP-ECJ which retrofitted grid-based EC software for use on the cloud. Instead, FlexGP has been intended, from its outset, to run on the cloud, and has been designed to take full advantage of the cloud.

We focus on the cloud-oriented design aspects of FlexGP. It is designed with an understanding that the cloud supplies sufficient computational resources upon request, yet expects these resources might fail or be delivered with unknown latency. It is conceived as a long-running computational learner, evolutionarily adapting and continuously improving its model, whilst allowing for drastic changes in supplied cloud resources and topology. It is implemented as a collaboration of many heterogeneous FlexGP instances, independently learning a model and observing the topology of the network. CloudGPML is implemented on a private cloud running OpenStack, a free and open source software suite providing Infrastructure as a Service (IaaS) for clouds. To integrate timestamps across nodes, we rely on Network Time Protocol (NTP), standard on Ubuntu 12.04, to provide accurate time synchronization of the nodes. This is sufficient for our purposes, as FlexGP operates on the scale of many seconds to minutes, and is not affected by microsecond variations.

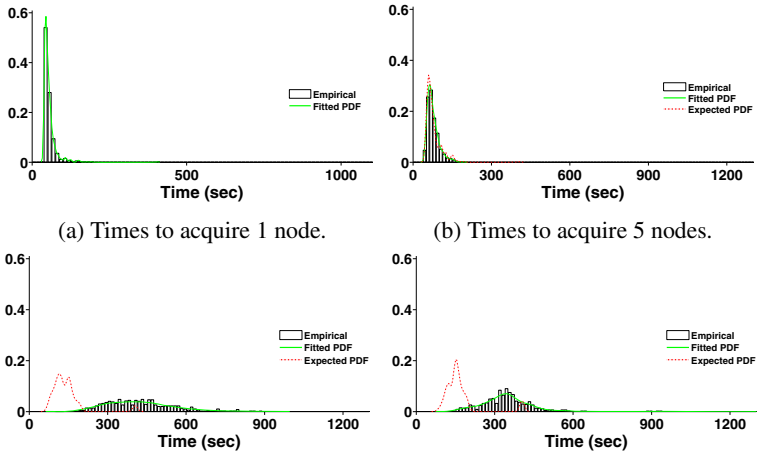
FlexGP-ECJ used a centralized master to coordinate starting GPML on nodes and establishing a network topology. This caused severe bottlenecks in starting the system and introduced vulnerabilities to node failures. In FlexGP, there is no single controller coordinating the system. It launches via a unique parallel asynchronous startup protocol. Integrated within this protocol is a stochastic factorization of the GPML parameters, creating a heterogeneous network of diverse learners. FlexGP runs a completely decentralized *gossip* neighbor discovery protocol at its IP layer. The protocol establishes the network simultaneously with the cascaded launch and integrates new instances into the network. It is resilient to instance failure and allows communication to continue even when instances disappear. Finally, it also enables interim results collection.

We proceed in the following manner: Sect. 2 describes the asynchronous launch procedure of FlexGP. Sect. 3 describes how each GPML learner is started with different data and parameters, ensuring a diverse set of models to support ensemble learning. Sect. 4 describes its IP discovery protocol. Recognizing that it is easier to compare FlexGP to other approaches after it has been described, a discussion of related work is deferred to Sect. 5. Sect. 6 concludes.

2 Parallel Asynchronous Startup

Applications typically request instances from a cloud in batches. The cloud possibly queues these batch requests and may decompose them; interleaving them with requests from other users. This might depend on batch size or the cloud's use of an internal fine-grained queue and a scheduler. Regardless of what a particular cloud does, the instance scheduler implementation should be treated as opaque by application designers.

In designing FlexGP's launch protocol, we started by studying the severity of latency in acquiring cloud instances. We assume that the time elapsed between requesting an instance and when that instance has booted and begins running our code, the *latency*,



(a) Times to acquire 1 node.

(b) Times to acquire 5 nodes.

(c) Times to acquire 50 nodes.

(d) Times to acquire 100 nodes.

Fig. 1. Probability distribution functions (PDF) of times to acquire nodes

is modeled by some distribution $P(u)$. We first estimated $P(u)$ by acquiring a single instance 1,000 times and measuring the latency, u, of each request. The data and its distribution are reported in Fig. 1a. If we optimistically assume a batch request of n instances is served in parallel as n independent requests by the scheduler, then the total latency, v_n, of the request ought to be the maximum of n independent samples drawn from $P(u)$. We estimated $P(v_n)$ for $n \in [5, 50, 100]$ from 500 samples and then fit a non-parametric distribution to the data. We report the observed data and fitted distributions alongside the predicted distributions (based on our measured $P(u)$) in Fig. 1. While the predicted and empirical distributions for $P(v_5)$ are close, the actual latency distributions for $P(v_{50})$ and $P(v_{100})$ are significantly larger than predicted.

This discrepancy indicates that smaller batch requests achieve closer to optimal latency than larger requests, and so our system ought to emphasize small batch requests over large ones. Futher, because acquiring many (50 or 100) instances may take significantly longer than acquiring the first 10 instances, we should start running GPML on an instance immediately after it boots, long before the entire set of nodes is acquired. Another concern when computing using the cloud is failing nodes. Requested nodes may never be acquired and running nodes may fail. This necessitates an architecture which is resilient to failures.

FlexGP implements a decentralized, peer-to-peer (P2P) startup algorithm in light of these observations. Every FlexGP instance is capable of launching other FlexGP instances. Immediately after booting, every FlexGP instance retrieves parameters from the node which started it. The parameters $\Psi.k$ and $\Psi.p$ indicate the number of nodes to start and the target IP list size (see Sect. 4), respectively. The GPML meta-parameters, Π, are used to determine the parameterization of each GPML learner (see Sect. 3). These steps are detailed in the NODESTART function in Algorithm 1.

Figure 2 left illustrates how FlexGP would launch 7 instances when $\Psi.k = 2$. Node A is launched and runs NODESTART(7, []), where [] indicates an empty list. A then

Algorithm 1. NODESTART(n, R)

n: nodes to launch, R: list of ancestor IP addresses
Ψ: launch parameters, Π: GPML meta-parameters
$ip \leftarrow$ LAST(R)
RETRIEVE(ip, Ψ, Π)
$R \leftarrow$ CAT(R, MYIP())
$n \leftarrow n-1$
if $n \leq \Psi.k$ and $n \geq 1$ **then**
  for $i = 1$ to n **do**
    $c_i \leftarrow$ BOOTNODE($1, R$)
else
  for $i = 1$ to $\Psi.k$ **do**
    $k \leftarrow \lfloor \frac{n}{\Psi.k-i+1} \rfloor$
    $c_i \leftarrow$ BOOTNODE(k, R)
    $n \leftarrow n-k$
IPDISCOVERY(R)
GPMLCOMPUTE()

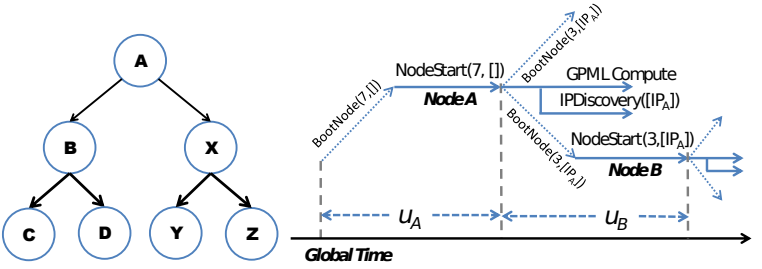


Fig. 2. A view of the launch of FlexGP for 7 nodes. Left: An initial node is launched and it brings up 2 more, which in turn bring up 2 more each, in a cascading fashion. Right: Timeline of booting and launching of instances. After starting more nodes, node A begins computation.

boots nodes B and X, each of which will run NODESTART(3, $[IP_A]$), and will go on to boot 2 more nodes each. Figure 2 right details the timeline of two nodes during startup, illustrating the concurrency present in the FlexGP startup. As soon as node A finishes executing NODESTART and started nodes B and X, it starts a new thread to begin running GPML computation and then continues into the IPDISCOVERY algorithm, as described in Sect. 4. This enables us to run GP concurrent with IP discovery and network discovery.

Figure 3 reports results from our experiments; demonstrating the advantages of the concurrent nature of FlexGP. The plot on the left shows the total progress of GPML as measured by individuals evaluated as global time progresses on the 150 nodes. The figure on the right examines the distribution of completed GP generations across all started nodes as the last node starts (around the 1800th second). The cumulative effect of this is that by the time the last node starts, some nodes have completed as many as 30 generations and some 2.2 million individuals have been evaluated across the FlexGP system.

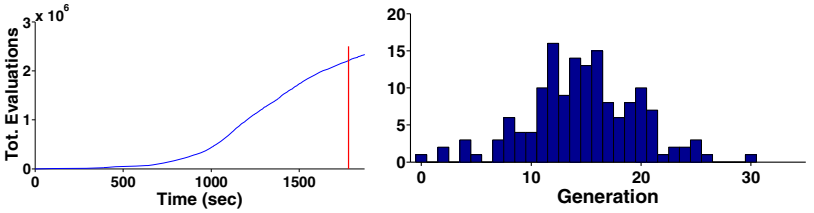


Fig. 3. Progress of GP on each node as distributed startup and IP discovery progress. Left: Cumulative fitness evaluations completed by all FlexGP nodes as launch proceeds. Right: Histogram of number of generations finished before the *time of last launch* (marked as red line on left figure).

Since each evaluation requires a pass through the training cases, this corresponds to at least 2.2 million passes through the problem dataset.

In Fig. 4 we compare how long it takes to start up 50 nodes for $\Psi.k \in [2,4,8,16]$. As expected, the time decreases as $\Psi.k$ increases, until $\Psi.k = 8$. Then the time gets worse for a value of 16. This is likely due to the wider variation in latencies for larger batch request sizes, as seen in Fig. 1. Note that the specific tradeoff point at $\Psi.k = 8$ is largely dependent on the properties of our cloud and the load it is under at the time of measurement and we expect this point would change over time or if measured on a different cloud.

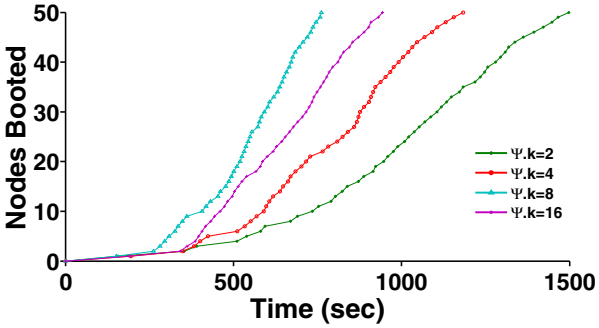


Fig. 4. Time to acquire 150 node as we change $\Psi.k$. Values reported are averages taken over 30 trials at each value.

The protocol is tolerant of node failures: the failure of one node interrupts the acquisition of further instances by that node, but does not hinder launches by other running nodes. For example, in Fig. 2, if node X failed to launch properly, nodes Y and Z will never be requested, but there is no affect on the acquisition of nodes B, C or D. In general, while the actual number of acquired nodes may not meet the requested N, GPML (and IP discovery) can execute on all nodes that have been acquired. We have taken the view that N will usually be large enough and failure will be sufficiently infrequent that we do not need to be concerned about any reporting, tracking and explicit recovery of node failures.

There may still be cases where the launch did not acquire a sufficient proportion of N instances. This may occur in the unlikely event that a node crashes very early on in the launch or in the face of intermittent cloud service interruptions. If such a scenario arises, we can simply tap an existing node and have it run the startup with new parameters which will try to populate the network with more resources. This same strategy can also be used to increase the number of running instances after startup. We might want to do this at night, when cloud instances become cheaper to run.

3 Factored Learners

The availability of massive on-demand expandable compute resources in the cloud enables us to learn many models in parallel. Bagging, boosting or simple parameter search are now feasible at a scale never seen before. FlexGP generates a large set of diverse models for ensemble learning. This is achieved by varying the parameters each GPML learner starts with. This leads to a set of factored learners, working in parallel to learn a diverse set of models.

We define the parameters for a GPML learner as $\{L, O, D, F\}$. L is the operator set provided to GP. O is the objective function used for normalizing fitness evaluations in GP. D is the set of training cases presented to the learner and F is the set of data features used in each training case in D. Note that while L and O are specific to GP, D and F are generic parameters of the data. Different options are available for these parameters, which are summarized in Table 1.

Table 1. GPML parameters and their possible values and definitions

Parameter	Value	Definition
Operator Set (L)	W	$\{+, -, /, *\}$
	X	$\{exp, ln\}$
	Y	$\{sqrt, x^2, x^3, x^4\}$
	Z	$\{sin, cos\}$
Objective Function (O)	Norm	Mean absolute error
	Norm-2	Mean squared error
	Norm-inf	Max error
Training Cases (D)	n	Subset of training cases, of size n
Feature Set (F)	m	Subset of features, of size m

We take Π, the set of meta-parameters retrieved from the parent (see Sect. 2), to define distributions over these options, guiding how FlexGP selects the values for each parameter. We will use the notation $p(O)$ to represent the distribution over the options for O. $p(L)$ gives probabilities for each of the 8 possible values of L.[2] $p(O)$ gives probabilities for each of the 3 *Norm*s defined. L and O are each chosen as a

[2] These values being W, $W \cup X$, $W \cup Y$, $W \cup Z$, $W \cup X \cup Y$, $W \cup X \cup Z$, $W \cup Y \cup Z$, and $W \cup X \cup Y \cup Z$.

single sample from $p(L)$ and $p(O)$, respectively. $p(D)$ defines probabilities of selecting each training case (for the set of all training cases). D is constructed by sampling without replacement n times from $p(D)$. The distribution for F is split into two parts. $p_1(F)$ gives probabilities for the number m of features to use. $p_2(F)$ defines the probabilities of using each feature. F is constructed by sampling m from $p_1(F)$ and then drawing m samples from $p_2(F)$ without replacement. For example, one learner could use the parameters $\{W, Norm2, \{x_1, x_2\}, \{d_{1...3000}\}\}$, while another learner might use $\{\{W \cup X\}, Norm2, \{x_1, x_4\}, \{d_{1...3000}\}\}$.

In our current implementation, we set all distributions in Π to be static, uniform distributions, for simplicity. However, since Π is retrieved from the parent node, the distributions over parameters may change dynamically as new nodes are launched. For example, a node could modify $p(D)$ to decrease the chance their children (the nodes it started) will select the same training cases it did. Or if new nodes are to be brought up after the system has been running, $p(D)$ could be modified to weight difficult training cases more and $p(F)$ could be modified to weight features with little apparent informative power less. We hope to explore such possibilities in the future.

4 Distributed IP Discovery

A key requirement of any cloud-based ML application is the support for communications between learners. Further, cloud-scale systems need an established network to robustly extract information and results from the system. For GPML, such communications might include the migration of individuals (models) or data-dependent summaries. Our work focuses on the establishment of such a network, but not on how the GPML learners use it.

The commonplace centralized architecture for network establishment (the so-called "master-slave" model) is not sufficient to meet the requirements of a cloud-based compute system [9]. In such an architecture, the nodes cannot begin computing until they receive parameters and IP lists from the master. This allows for the creation of arbitrary network topologies by the master. However, on a cloud the master cannot know the IP addresses until all instances are acquired. As we observed in Sect. 2, the latency for a many-node acquisition is quite large. Further, some of the requested nodes might fail before reporting to the master, complicating matters further.

To avoid this latency while still achieving the networking requirements of FlexGP, we design a distributed IP discovery protocol. Note, we focus here on the initial bootstrapping of the network - the "IP discovery" problem. This is separate from the problem of creating particular topologies in P2P networks [4]. Recall that as part of startup a parent node shares its IP list with all its children. A node at level i therefore has i IP addresses at startup. We then use a *gossip* protocol to populate the neighbor list at each node. First, we set a lower limit, $\Psi.p$, for the number of IP addresses a node needs to acquire. It generally is a function of the total number of nodes. We then follow an address passing protocol per Algorithm 2. In this protocol's active phase, each node selfishly tries to increase its IP addresses up to its limit by requesting more IP addresses from its neighbors while it shares with its neighbors its IP addresses in exchange. After it meets or exceeds the limit, in its passive phase, it serves any request it receives in exchange for their IP addresses.

Algorithm 2 IPDISCOVERY(R)

$\Lambda \leftarrow R$
loop
  $\lambda \leftarrow$ set of new messages received
  for m in λ **do**
    if $m.type$ is REQUESTIPLIST **then**
      $\Lambda \leftarrow$ MERGE(Λ, $m.\Lambda$)
      RESPONDIPLIST($m.ip$, Λ)
    else if $m.type$ is RESPONDIPLIST **then**
      $\Lambda \leftarrow$ MERGE(Λ, $m.\Lambda$)
  if LEN(Λ)$< \Psi.p$ **then**
    $\varepsilon \leftarrow$ RANDOM(Λ)
    REQUESTIPLIST(ε)

4.1 Empirical Study

We ran launch and IP discovery experiments with $N = 150$, and $\Psi.p = 25$. The plots of Fig. 5 show global time progressing along the x-axis. In Fig. 5 left, the y-axis denotes the number of IP addresses each node acquires. Each plotline is a node and, for clarity, we only show a subset of N. We observe that all nodes eventually acquire IP addresses of a significant number ($> \frac{N}{2}$) of nodes and in some cases almost all other nodes. A plotline changes from solid to dashed when a node switches from active to passive phase. It is interesting that the latter nodes acquire addresses of a very large number of nodes by gossiping with just one or two nodes arbitrarily. Interestingly, the last node acquires as many as 150 addresses. This ensures connectivity if we add more nodes later. Finally, note that all nodes have at least $\Psi.p$ nodes before the last one is booted.

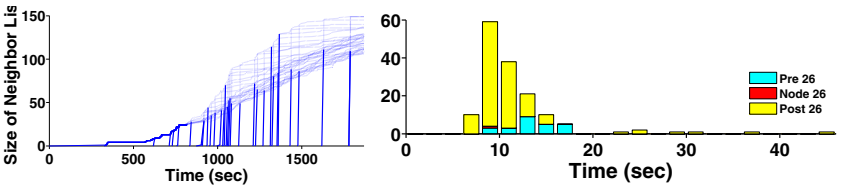


Fig. 5. IP discovery through gossip. Left: Progress of IP discovery as a function of global system time. Each line represents the number of IP addresses a node accumulates as time progresses. Right: Time it took for each node to acquire $\Psi.p$ IP addresses.

Figure 5 middle shows the distribution of delays for nodes to enter the passive phase of IP discovery. This delay for the i^{th} node is calculated as follows. Let S_i be the time at which the i^{th} node started and let T_i be the time at which the i^{th} node discovered its 25^{th} IP address. Then the latency for node i is given as $T_i - max(S_i, S_{26})$. We notice that almost 130 nodes take less than 25 seconds to enter the passive phase.

5 Related Work

There is a large body of distributed EC research which focuses exclusively on the design of distributed, algorithmic models, like island-based GP, instead of designs which take advantage of a particular resource type or communication layer. Much of this work is only tangentially related to FlexGP, as we are focused on writing an EC platform which takes advantage of the cloud platform. The systems in [1, 3, 10–12] rely upon MapReduce for parallelization. MapReduce is a powerful platform for distributed computation, but its dependence upon a separate distributed file system, single point of failure in the master and synchronization bottlenecks are not a good match with the cloud.

FlexGP's IP discovery is like other EC peer-to-peer systems. For example, the EvAg system [5, 6] also relies upon gossiping for node discovery. Little information is available on its startup method. It is not specialized to run on particular resource types whereas it is designed to investigate topology and a fine grained distribution model. EvAg and FlexGP differ in how they introduce evolutionary diversity: EvAg employs different operators across randomized neighbourhood whereas FlexGP factors each island with differentiation of data, GP objective function, operator set and input variables. Folino introduced peer to peer based design for building classifier ensembles [2].

6 Conclusions and Future Work

In this paper we have described the development of FlexGP, a large-scale, distributed system using GP for machine learning on the cloud. Our goal was to establish a large network of independent GPML leaners on the cloud with an eye towards minimizing latencies and bottlenecks while maximizing resource utilization. FlexGP is an improvement over its precursor FlexGP-ECJ, avoiding its bottlenecks in starting up and establishing a network topology. For a modest sized experiment of 150 nodes attempting to solve a small scale regression problem using FlexGP we showed the efficiency gains by introducing parallelization schemes and concurrency at multiple levels.

We plan to continue development on FlexGP, refining our platform and adding new features. The elegant design of this framework allows seamless expansion of the computation as needed. We intend to use this feature in several ways. If more resources become available, we would like to bias the distributions in Π for the new nodes based on insights from the running computation. We would also like to introduce multiple levels of parallelism to solve big data problems. Each FlexGP node can start a FlexGP system of its own, running an island-based GP with migration in a fully connected network. Additionally, we will develop shared memory, multicore parallelization at each GPML learner. This will allow us to run bigger population sizes and solve larger problems.

Acknowledgements. This work was supported by the GE Global Research center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of General Electric Company.

References

1. Fazenda, P., McDermott, J., O'Reilly, U.-M.: A Library to Run Evolutionary Algorithms in the Cloud Using MapReduce. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 416–425. Springer, Heidelberg (2012)
2. Folino, G., Forestiero, A., Spezzano, G.: A jxta based asynchronous peer-to-peer implementation of genetic programming. Journal of Software 1(2), 12–23 (2006)
3. Huang, D.W., Lin, J.: Scaling populations of a genetic algorithm for job shop scheduling problems using MapReduce. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 780–785 (December 2010)
4. Jelasity, M., Montresor, A., Babaoglu, O.: T-Man: Gossip-based fast overlay topology construction. Computer Networks 53(13), 2321–2339 (2009); Gossiping in Distributed Systems
5. Jiménez Laredo, J.L., Lombraña González, D., Fernández de Vega, F., García Arenas, M., Merelo Guervós, J.J.: A Peer-to-Peer Approach to Genetic Programming. In: Silva, S., Foster, J.A., Nicolau, M., Machado, P., Giacobini, M. (eds.) EuroGP 2011. LNCS, vol. 6621, pp. 108–117. Springer, Heidelberg (2011)
6. Laredo, J., Eiben, A., Steen, M., Merelo, J.: Evag: a scalable peer-to-peer evolutionary algorithm. Genetic Programming and Evolvable Machines 11, 227–246 (2010)
7. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed GraphLab: A framework for machine learning and data mining in the cloud. Proc. VLDB Endow. 5(8), 716–727 (2012)
8. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in Action. Manning Publications Co. (2011)
9. Sherry, D., Veeramachaneni, K., McDermott, J., O'Reilly, U.-M.: Flex-GP: Genetic Programming on the Cloud. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 477–486. Springer, Heidelberg (2012)
10. Verma, A., Llora, X., Goldberg, D., Campbell, R.: Scaling genetic algorithms using mapreduce. In: Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, pp. 13–18 (December 2009)
11. Verma, A., Llora, X., Venkataraman, S., Goldberg, D., Campbell, R.: Scaling eCGA model building via data-intensive computing. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (July 2010)
12. Wang, S., Gao, B.J., Wang, K., Lauw, H.W.: Parallel learning to rank for information retrieval. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, pp. 1083–1084. ACM, New York (2011)

Cloud Scale Distributed Evolutionary Strategies for High Dimensional Problems

Dennis Wilson, Kalyan Veeramachaneni, and Una May O'Reilly

Massachusetts Institute of Technology, USA
{dennisw,kalyan,unamay}@csail.mit.edu

Abstract. We develop and evaluate a cloud scale distributed covariance matrix adaptation based evolutionary strategy for problems with dimensions as high as 400. We adopt an island based distribution model and rely on a peer-to-peer communication protocol. We identify a variety of parameters in a distributed island model that could be randomized leading to a new dynamic migration protocol that can prove advantageous when computing on the cloud. Our approach enables efficient and high quality distributed sampling while mitigating the latencies and failure risks associated with running on a cloud. We evaluate performance on a real world problem from the domain of wind energy: wind farm turbine layout optimization.

1 Introduction

Our goal is to design Estimation of Distribution Algorithms, or EDAs, for high dimensional problems via large scale cloud computing resources: hundreds or even thousands of cores made available via virtualization of commodity hardware. Cloud-scaling helps us achieve the much needed higher sampling rates for high dimensional problems. While the cloud provides us access to large number of resources on demand, resource sharing through virtualization implies that some nodes could be slower than others. Hence algorithms designed based on synchronous computation/communication and shared, distributed memory architectures could incur latencies. We are investigating whether a certain type of distribution model for EDAs is more amenable to these on-demand resources and what distribution protocol and software infrastructure we should build to support them.

An example model is a master-slave fitness distributed model with a very large sample population. However, in this model a bottleneck arises when the distribution is re-estimated since the entire population needs to be evaluated before a sub-sample is selected to re-estimate the distribution. Recognizing this, as well as due to success of island based models in classical evolutionary algorithms, we are exploring an island based model where multiple instances of an EDA, one per island, optimize locally while periodically communicating progress information to neighbors.

The asynchronous execution and communication between islands allows for desired higher sampling rates, but requires a distribution methodology. We map

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 519–528, 2013.
© Springer-Verlag Berlin Heidelberg 2013

each island to an independent node (with either single or multiple cores) and build a socket level communication layer across the network of nodes. In this submission we present a cloud EDA algorithm we have named CASINO– **C**loud **A**ssets for **S**tochast**I**c **N**umerical **O**ptimization. CASINO is a communication framework for EDA development that is used here to distribute the **C**ovariance **M**atrix **A**daptation based **E**volutionary **S**trategy, CMA-ES [1].

In designing a distributed CMA-ES, we focus on the communication protocol for the migration of progress information. We examine whether randomized migration protocols are effective when compared to static and centralized protocols, as well as the type of information exchanged by the independent EDAs. Conventionally island models communicate current best solutions. In this paper we experiment with CMA-ES passing either the best solutions or the island's covariance matrix.

Our evaluation perspective is practical; we focus on a real world wind farm turbine layout optimization. We proceed as follows: Section 2 briefly considers related work. In Section 3 we briefly review the CMA-ES algorithm. Section 4 describes CASINO. In Section 5 we describe different randomized protocols for sending information between islands. Section 6 describes the layout optimization problem we use CASINO for. Section 7 compares different strategies based on their performance on our exemplar problem. Section 8 concludes with future work.

2 Related Work

There is a large body of literature on methods of distributing Evolutionary Algorithms, EAs, for which [2–5] serve well as overviews. In some circumstances, simple parallelization models such as independent, parallel runs or master-slave fitness evaluation suffice. There are examples of continuously valued, distributed EAs for numerical optimization which commonly use Particle Swarm Optimization, Evolutionary Strategies, or Genetic Algorithms [6–11]. A few of these approaches focused on adapting MapReduce to scale algorithms to compute resources on the cloud [10].

A closely related work appears in [7] where authors have used a Message Passing Interface, or MPI, over a distributed file system. They developed this system for compute grids and achieved efficiencies via MPI over distributed, shared and hybrid memory systems. In our current work we do not use or require such a file system since we believe that it can cause latencies and is not particularly required for an island based model which has asynchronous and infrequent communications. Additionally, on the cloud we cannot assume the multiple cores of our virtual machines reside on the same physical machines, which algorithms based on shared memory systems assume.

3 Distributed CMA-ES Strategy

CMA-ES self-adapts the covariance matrix of a multivariate normal distribution. This normal distribution is then sampled to draw the variables of a candidate

solution in the multidimensional search space. The covariance matrix guides the search by adaptively biasing sampling toward historically profitable correlations between the variables. This makes the evolutionary search powerful.

Consider a representation $\mathbf{x}_k$ for the k^{th} solution to the optimization problem that attempts to minimize the objective function $f(\mathbf{x})$. In each iteration, t, the algorithm samples λ number of solutions from a multivariate normal distribution given by

$$\mathbf{x}_k^{(t+1)} \sim \mathcal{N}(\mathbf{m}^{(t)}, \sigma^{2(t)}, \mathbf{C}^{(t)}) \forall k. \tag{1}$$

After evaluating these solutions against the fitness function a subset μ are selected for updating the mean and covariance of the multivariate distribution. The mean is updated by

$$\mathbf{m}^{(t+1)} = \sum_{i=1}^{\mu} \omega_i \mathbf{x}_i^{(t+1)}, \text{such that} \sum_{i=1}^{\mu} w_i = 1 \text{and} w_i > 0 \tag{2}$$

The covariance matrix could be simply updated by:

$$\mathbf{C}_{\mu}^{(t+1)} = \sum_{i=1}^{\mu} w_i \left(\frac{\mathbf{x}_i^{(t+1)} - \mathbf{m}^{(t)}}{\sigma^{(t)}} \right) \left(\frac{\mathbf{x}_i^{(t+1)} - \mathbf{m}^{(t)}}{\sigma^{(t)}} \right)^T \tag{3}$$

The CMA-ES algorithm also incorporates additional information based on trajectory of the mean and the covariance matrix as iterations progress. For further information we refer readers to [1].

4 CASINO Setup

One of the main goals in CASINO's design was a simplistic communication protocol and the ease of its setup. The network layout and migration protocols can be easily manipulated by changing neighbor selection methods, information passed, and communication frequency without disturbing the underlying architecture. The Instrument Control Toolbox in MATLAB is used to facilitate information passing over TCP-IP, and system-level `ssh` calls are used for notification. CASINO's setup has the following configuration steps:

Step 1: Infrastructure: A central server node requests a batch of nodes from the cloud. The server collects, and then broadcasts, the nodes' IP addresses as list R. A node i establishes a connection with another node j from its IP list R_i by creating an exclusive mailbox with a unique id. The mailbox configuration is determined by the user; while we have explored different migration topologies, every node has the capacity to engage in various topologies given R_i. Each node also creates a *notification* mailbox which a sender uses to inform it that a message has arrived at the sender's exclusive mailbox.

Step 2: Information Communicated: Currently, we allow each CMA-ES node to send either a subset of its individuals or its covariance matrix. In the

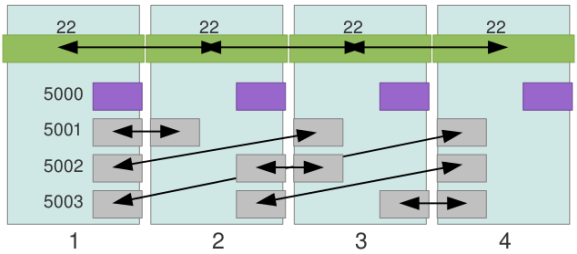


Fig. 1. An example of a 5-node network. The server node (not shown) is connected to port 5000 on each worker node. These nodes are fully connected to each other using the ports numbered higher than 5000. They also all use port 22 as a notification mailbox.

case of former, the best μ individuals are chosen, though the entire set could be communicated. The fitness of the individuals is sent as well to avoid reevaluation. Similarly, covariance matrices are reduced to the upper triangle of the matrix before passing and the best individual's fitness is included to combat redundancy. We are also interested in communicating parts of a larger, centralized covariance matrix, but have not yet explored this option.

Step 3: Migration Protocol: The details of the migration protocols will be fully discussed in Section 5. Our ability to experiment with different migration protocols comes from the infrastructure configured in Step 1 as well the nature of CMA-ES algorithm where information received can be integrated in any of its iteration. We experiment with a variety of pre-determined migration protocols and a set of randomized strategies.

Step 4: Message Information Integration: During a run, a node integrates the information it receives from its neighbors. When the information unit is its neighbor's best μ solutions, it simply merges these into its population before making a selection of best μ from which the next covariance matrix is estimated and subsequently updated.

When the unit is the entire covariance matrix, we modify the CMA-ES algorithm include the covariance matrix from the neighbor:

$$\alpha(1 - c_{cov})\mathbf{C}^{(t)} + (1 - \alpha)(1 - c_{cov})\mathbf{C_n}^{(t)} \tag{4}$$

where α is the relative weight. We call this *neighbor-update*. When covariance matrices arrive from multiple nodes, we rank the covariance matrices based on their associated fitness, that of the best population on their island, and choose the best one to integrate.

5 Randomized Migration Protocols

Next we attempt to overlay different migration protocols on our infrastructure. A migration protocol is defined by three aspects: topology chosen, topology

parameters, and type of information. Below we describe different ways one can make choices in these.

Topology Selection: With regards to topology one can overlay a fixed topology like *Ring*, *Broadcast*, or even *No communication* which we call a *static* topology. An alternative is to choose a topology at random every generation. This is possible because each node in our network has the IP addresses of all the other nodes. This we call a *dynamic* strategy.

Parameter Selection: There are two parameters that are specified in a migration protocol. They are: q, number of neighbors, and γ, migration frequency. Selection of these parameters can have significant influence on the network use and possibly latencies. These parameters are chosen randomly as per a probability distribution; in our case $\gamma \sim U[5, 10]$ and $q \sim \mathcal{N}(k, \frac{k}{2})$ where $k \in [n, \frac{n}{2}]$. Our current framework allows the user to choose these parameters in the following ways: the choice can be made centrally and every island is passed the same parameters at initialization (homogeneous) or each island can choose its own parameters based on the distribution (heterogeneous). Additionally, we allow for further protocols by introducing randomization during run time (RR). Each nice can change its parameter q during run time and decide whether or not send randomly by flipping a biased coin.

A user of our framework can select any configuration in terms of topology and parameters. We experimented with a few and the table below presents the names we use for these protocols and the choices that are made for topology and parameters in each of them.

Table 1. Different protocols tested. HO implies homogeneous, HE implies heterogeneous, and RR implies randomized during run time

Protocol	**Topology**	**Parameters**	
		q	γ
Static	Static	HO	HO
Static-Random Frequency	Static	HO	HE
Dynamic	Dynamic	HE	HE
Dynamic-Random Frequency	Dynamic	RR	HE
Dynamic-REG	Dynamic	RR	RR

6 An Exemplar High Dimensional Problem

To analyze the performance of the algorithm under a variety of choices, we selected a real world high dimensional problem. We chose a wind energy layout problem that has been studied by a number of researchers who have applied either centralized version (with multithreading for parallelizing fitness evaluation) of the CMA-ES algorithm. The goal is to identify a turbine layout, given by the x, y co-ordinates for the turbines, that maximizes the energy capture from a given farm

$$\arg\max_{(X,Y)} \eta(X, Y, v, \beta(v)) \tag{5}$$

where v is the wind speed, and the function $\beta(v)$, known as a power curve, gives the power generated by a specific turbine for a given wind speed. Wind speed v however is a random variable with a Weibull distribution, $p_v(v, c, k)$, which is estimated from wind resource data. This distribution also changes as a function of direction, θ which varies from $0^0 - 360^0$, yielding a probability density function for different θ given by $p_v^\theta(v, c, k)$. Additionally, wind flows from a certain direction with some probability $P(\theta)$. These different pieces of information are inputs to the algorithm. Due to the random nature of wind velocity, the objective function evaluates the *expected* value of the energy capture for a given wind resource and turbine positions. For a single turbine, this value can be calculated using

$$E^i[\eta] = \int_\theta P(\theta) \int_v p_v^\theta(v, c_i, k_{,i}\, x_i, y_i, X, Y)\beta^i(v). \tag{6}$$

Equation 6 evaluates the overall average energy over all wind speeds for a given wind direction, and then averages this energy over all the wind directions. c_i, k_i are turbine specific resource parameters derived for the i^{th} turbine after wake calculations. For more details, refer to [12].

The goal of the optimization problem is to maximize Equation 6. This problem has analysis value because of its high dimensionality, non-linear variable relationships and expensive fitness evaluation.

7 Experiments and Analysis

We use a 200 turbine problem with a 400 dimensional search space for evaluation. All experiments use 100 cloud nodes with an island on each. The CMA-ES parameters of each island are $\mu = 10$ and $\lambda = 20$. All experiments are run 20 times and the results presented are averages. Each run takes approximately 18 minutes. The fitness evaluation for a 200 turbine problem takes **5** seconds.

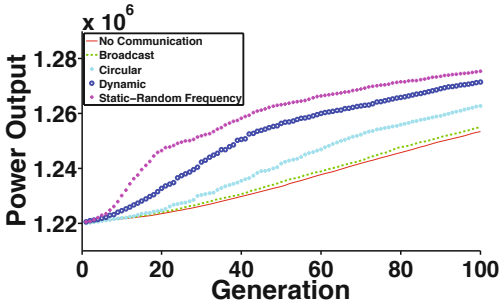
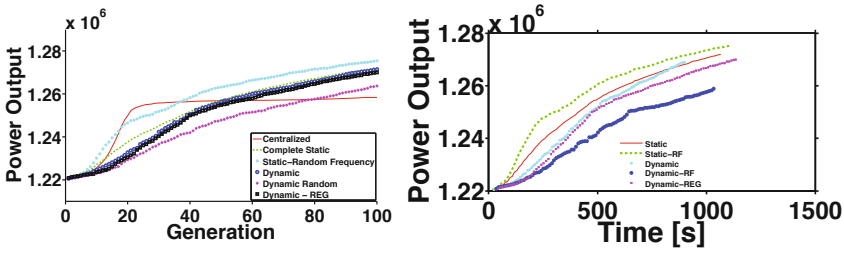


Fig. 2. Performance of different static topologies vs randomized topologies

Are Dynamic Topologies Harmful?
We first compare *Static-Random Frequency* and *Dynamic* which have random migration rates and random topologies to *Static* with *Ring* and *Broadcast* topologies. For fair comparison, *Static-Random Frequency* and *Ring* both have $q = 2$ implying they exchange the same amount of information. In *Dynamic* the number of neighbors is drawn randomly $q \sim \mathcal{N}(k, \frac{k}{2})$ at the beginning of the run, where $k = 2$. We include an experiment of *No communication* also.

Per Figure 2, as expected, *No communication* fares worst. *Broadcast* was as poor or statistically the same as *No communication* likely because there was too much information exchanged.The randomized strategies work better than *Ring* suggesting that a random topology is at least not harmful, and might drive advantageous population diversity.



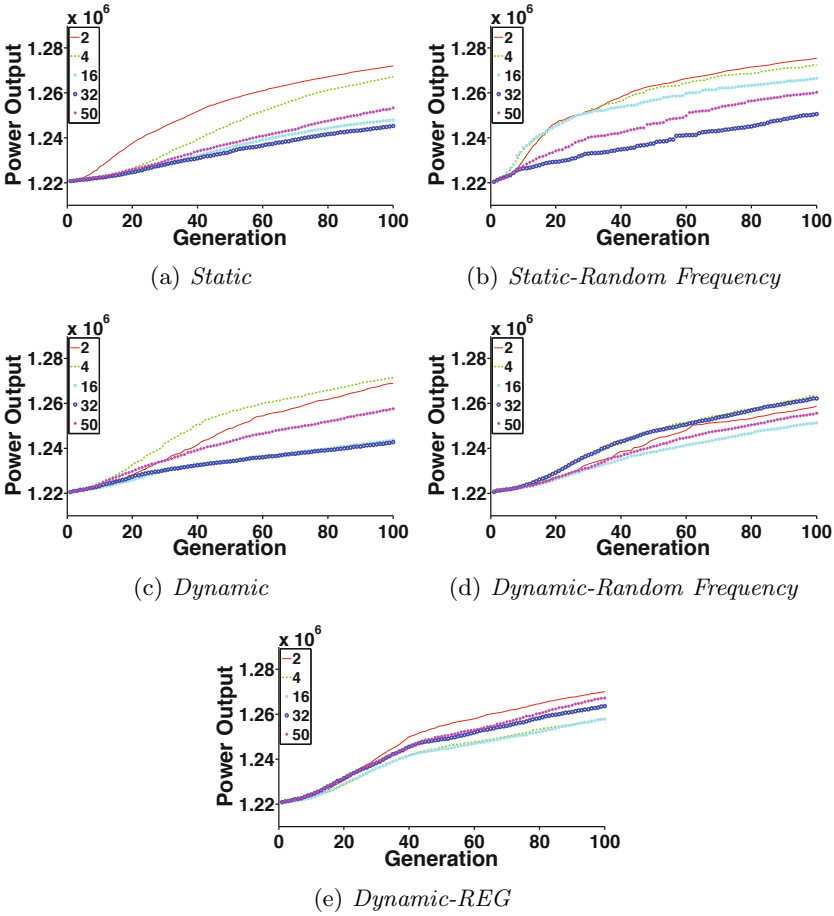
(a) (b)

Fig. 3. Comparing centralized CMA-ES with 100 times the population to a distributed CMA-ES with smaller populations and different information sharing protocols. The time scale is also show as a performance comparison; the centralized CMA-ES runtime was of a higher order of magnitude and is not shown.

Are Islands Harmful?
As another check, we compare the island model to a CMA-ES where all samples are *centralized*, ($\mu = 1000$ and $\lambda = 2000$) and each sample directly affects the covariance matrix update. We observe that the *centralized* CMA-ES prematurely converges quickly (within 25 generations). Initially, it outperforms the distributed protocols, but within 100 generations, all of them surpass it. The initial benefit of *centralized* CMA-ES is its complete communication and instantaneous integration of results. In the island model, full integration hangs on result migration.

This clearly shows that distributed sampling is not harmful and could, indeed, be advantageous. We also compare the time taken for different protocols to finish. Figure 3(b) shows the progression of different approaches in terms of fitness as time progresses. All values are averaged over 20 trials.

What Is the Best Number of Neighbors? We now evaluate the sensitivity of each distributed strategies and randomized protocols to q, the number of



(a) *Static* (b) *Static-Random Frequency*

(c) *Dynamic* (d) *Dynamic-Random Frequency*

(e) *Dynamic-REG*

Fig. 4. Comparison of q for distributed protocols

neighbors or, for dynamic protocols with respect to q, the mean of the distribution from which the number of neighbors was determined. Figure 4 shows each protocol with $q = 2, 4, 8, 16, 32, 50$. The *Static* protocol is best with a two neighbor topology. The best q for the other randomized protocols is not discernible statistically, though, in the case of *Dynamic* $q = 4$ appears to best the others by a slim margin. Across the protocols, this makes it infeasible to state whether the "best" q remains the same or differs.

Static-Random Frequency performs the best overall, but was also the most costly in computation time, taking 23 minutes per run on average. This is caused by processing time of incoming and outgoing populations. While each node in this protocol sends its populations to the number of individuals indicated, 2 for the best run, a single node may be a receiver more frequently. The overburdened nodes increase the overall optimization time.

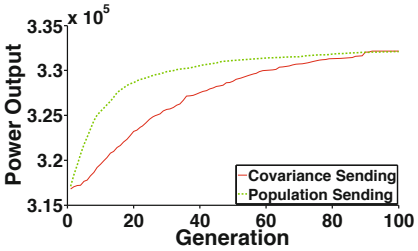


Fig. 5. Performance comparison of communicating best solutions vs the covariance matrix for a 50 turbine optimization problem using the *Static-Random Frequency* protocol

Static-Random Frequency and *Dynamic* both showed increased variability between values of q. Multiple runs of *Dynamic-Random Frequency* and *Dynamic-REG* outperformed the others but not on average. This is the impact of increased randomization; a randomized dynamic network may end up having little communication, or communication may be isolated to a small section of the population.

Information Unit: Individuals vs Covariance Matrix. Time permits us only to briefly study the difference between communicating the μ best individuals as units of information to exchanging the covariance matrix, see Figure 5. Note there will be a tradeoff point between μ and dimensionality where the message size of each will cross. A covariance matrix for an n-dimensional problem is of size $\frac{n^2}{2}$. For a 200 dimension problem this message size and update time is high, so we compared the two information units on a 50 dimension problem. The best individual information outperformed the covariance matrix approach initially, but by 100 generations both experiments achieve the same mean best fitness. This is likely due to our decision to only integrate the best covariance matrix received from neighbors each iteration. This slows the sharing of crucial information.

8 Conclusions and Future Work

In this paper we presented our island based CMA-ES algorithm capable of running on the cloud. We identified a variety of parameters for an island based model which can be randomized in order to overcome the latencies introduced by the cloud due to its virtualization layer and resource sharing. We investigated the performance of these strategies (on a real world problem) by running on our private cloud. We also investigated whether or not passing covariance matrix helps the distributed model in terms of performance. Passing the covariance matrix is extremely expensive and it is not clear from the brief study we performed if it is beneficial. We did observe that the pattern of convergence is different. As part of future work we would like to investigate this further.

This paper's investigation related to *dynamic and randomized migration for a cloud based black box optimization* is likely to generally extrapolate to similar island model versions of EDAs. This is because EDAs do not each individually

make special consideration related to topology, e.g. the source and destination of communicated information in CMA-ES or other such algorithms is not specific to the algorithm. For our investigation with respect to *what information to exchange*: when current best solutions are migrated, results should extrapolate to other algorithms. However, because the covariance matrix is not common to all approaches, the findings are restricted to CMA-ES.

Acknowledgements. Dennis Wilson acknowledges the support of MIT Energy Initiative. Una-May and Kalyan acknowledge the support from GE Global Research center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of General Electric Company or MITEI.

References

1. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J.A., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms, pp. 75–102. Springer (2006)
2. Alba, E.: Parallel metaheuristics: a new class of algorithms, vol. 47. Wiley-Interscience (2005)
3. Tomassini, M.: Spatially structured evolutionary algorithms. Springer (2005)
4. Nedjah, N., Alba, E., de Macedo Mourelle, L.: Parallel Evolutionary Computations. Springer (2006)
5. Cantú-Paz, E.: Efficient and accurate parallel genetic algorithms. Springer, Netherlands (2000)
6. Zhu, W.: Nonlinear optimization with a massively parallel evolution strategy pattern search algorithm on graphics hardware. Applied Soft Computing 11(2), 1770 (2011)
7. Müller, C.L., Baumgartner, B., Ofenbeck, G., Schrader, B., Sbalzarini, I.: pcmalib: a parallel fortran 90 library for the evolution strategy with covariance matrix adaptation. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1411–1418. ACM (2009)
8. Rubio-Largo, Á., González-Álvarez, D.L., Vega-Rodríguez, M.A., Almeida-Luz, S.M., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: A Parallel Cooperative Evolutionary Strategy for Solving the Reporting Cells Problem. In: Corchado, E., Novais, P., Analide, C., Sedano, J. (eds.) SOCO 2010. AISC, vol. 73, pp. 71–78. Springer, Heidelberg (2010)
9. Rudolph, G.: Global Optimization by Means of Distributed Evolution Strategies. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 209–213. Springer, Heidelberg (1991)
10. Gunarathne, T., Wu, T.L., Qiu, J., Fox, G.: Mapreduce in the clouds for science. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), November 30-December 3, pp. 565–572 (2010)
11. Verma, A., Llora, X., Goldberg, D., Campbell, R.: Scaling genetic algorithms using mapreduce. In: Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, November 30-December 2, pp. 13–18 (2009)
12. Kusiak, A., Song, Z.: Design of wind farm layout for maximum wind energy capture. Renewable Energy 35(3), 685–694 (2010)

Malicious Automatically Generated Domain Name Detection Using Stateful-SBB

Fariba Haddadi[1], H. Gunes Kayacik[2], A. Nur Zincir-Heywood[1], and Malcolm I. Heywood[1]

[1] Computer Science, Dalhousie University, Halifax, NS, Canada
{haddadi,zincir,mheywood}@cs.dal.ca
[2] Glasgow Caledonian University, Scotland, UK
gunes.kayacik@gcu.ac.uk

Abstract. This work investigates the detection of Botnet Command and Control (C&C) activity by monitoring Domain Name System (DNS) traffic. Detection signatures are automatically generated using evolutionary computation technique based on Stateful-SBB. The evaluation performed shows that the proposed system can work on raw variable length domain name strings with very high accuracy.

Keywords: Security, Botnet detection, Evolutionary computation, Data mining.

1 Introduction

In the world of fast growing Internet and online activities which almost everyone has something to share and benefit from, having a secure infrastructure is the primary need to protect users' identity and information. Due to the high reported botnet infection rate and its wide range of distributed illegal activities, botnets- among various types of malwares– are one of the main threats against the cyber security [1].

Every year new reports are published indicating that the number of botnet victims is increasing. In 2010, Damballa Inc. published a paper on the top 10 active botnets indicating that botnet infection rate is rapidly increasing by the average growth of 8% per week [2]. McAfee thread reports also confirm that this growth continues into 2012 [3]. These reports also indicate that new powerful botnets enter the Internet realm every year. Moreover, in response to improvements in detection mechanisms, botnets update themselves as well. From the security perspective, these observations indicate that knowing about a botnet mechanism and detecting it would not always be enough in a condition that the master has the opportunity to upgrade or even change its mechanism completely. In this situation, the botnet monitoring activity should be continuous and the botnet detection mechanism should also upgrade itself by the patterns learned through the monitoring process. In other words, this is an arms race and therefore, automating the detection mechanisms as much as possible will give the much needed headway to the defender side. Thus, in this research, we explore how far we can push to automatically generate signatures based on minimum a priori information in order to adapt to the changes in the botnet upgrades.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 529–539, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Monitoring network traffic at DNS level provides a suitable solution to mitigating botnet attacks because, in addition to its many legitimate uses, DNS can also be used by botnets to manage their infrastructure. In a typical botnet for example, the infected computer may locate the C&C server by querying a list of domain names, which are supplied at the time of infection or after. C&C server will instruct the infected host to engage in malicious activities, such as data ex-filtration, denial of service attacks or serving spam, without user's knowledge. The list of domain names provided to the victim host is large enough so that it cannot be blacklisted manually or at firewall level. Thus, to create a long list of domain names, attackers usually generate the list algorithmically. Generated domains exhibit structural and syntactical anomalies compared to regular domain names. It is therefore possible to detect botnet C&C activity by monitoring high volume access to unusually structured domain names.

To detect these anomalies, we employ an evolutionary computation technique based on SBB [4]. Our proposed system employ a modified version of SBB, hereafter called Stateful-SBB (Stateful Symbiotic Bid-Based Genetic Programming). Where most classification algorithms require features to be defined a priori (need behaviour analysis on botnets conducted by human resources), Stateful-SBB works on the raw domain name strings (no a priori information) and achieves comparable detection rates without requiring a predefined feature set. Avoiding such a requirement is the most important contribution of this paper since this enables the approach to adapt to the changes in the botnet upgrades. The remainder of the paper is organized as follows. Section 2 details Botnet topologies, detection methods and related works in this field. Our methodology and the proposed system are discussed in Section 3. Results are provided in Section 4 and conclusions are drawn in Section 5.

2 Related Work

In this section we will give an overview of how botnets work and the existing detection mechanisms in the literature.

2.1 Botnets: How They Work

A bot program is a self-propagating malware that infects vulnerable hosts known as bots (zombies) and is designed to perform a task after being triggered. The infected bots network is referred as botnet, which is under the remote control of a master called botmaster. Usually bots receive commands from the master through a C&C communication channel and carry out malicious tasks such as Distributed Denial of Service (DDoS), spamming, phishing and identity theft attacks [1] [5].

Unlike the earlier botnets that had a list of exploits to launch on targets and all the commands were set at the time of infection, today a typical advanced bot uses multiple phases to create and maintain a botnet including: initial infection, secondary injection, connection, malicious C&C, update and finally maintenance [1] [5]. In the first phase, attacker infects the victim using several exploitation techniques to find its existing vulnerabilities. Once the target got infected, in the second phase, the shell-code is executed on the victim machine to fetch the image of the bot binary which then installs itself on the machine. At this time, the host is completely converted to a

zombie and malicious tasks can run automatically on the host. In the connection phase, the bot binary establishes the C&C channel to be used by the master to send the commands to its bot army (botnet). Finally, when the master needs to update the bots for several reasons such as avoiding an antivirus, changing the C&C server setting, or adding a new functionality, the update and maintenance phase is entered.

It is believed that until 2003, most of the botnets were using centralized topology, utilizing IRC Protocol [6] [7]. Since 2003, not only botnets have started to use several protocols such as HTTP and DNS as well as the decentralized topology, but also they have started to employ fluxing methods to avoid detection. Fluxing is a technique used to move the communication between the victims and the C&C server from domain to domain using the DNS protocol [8]. Therefore, since 2004 DNS is used in botnets to add mobility and to remove the single point of failure [7].

2.2 Botnet Detection

Mainly, there are two approaches for botnet detection [1]. The first approach is based on honeynets. Honeynet-based techniques are mostly useful to realize botnets characteristics and technology but not necessarily detection.

The second approach is based on network traffic monitoring and analysis which are typically classified as: Anomaly-based, or Signature-based. Anomaly-based methods rely on finding network anomalies and unusual behaviors such as high volume of network traffic, which could be the outcome of botnet presence in the network. However correctly modeling the network normal behavior is challenging. On the other hand, signature-based methods create signatures to be used for detection purposes. Necessity of prior knowledge of botnets and their behavior make the detection systems of this kind vulnerable to unknown (new) botnets.

There are several detection mechanisms of the second approach that specifically focus on identifying malicious domain names, which are used by DNS-based Botnets. E. Stalmans et al. developed a system to detect fast-flux domains using DNS queries [9]. Analyzing the DNS query responses, two groups of features were extracted to identify legitimate and malicious queries: DNS and Textual features. Given the extracted features, they employed C5.0 and Bayesian classifiers to identify fast-flux queries. S. Yadav et al. proposed a system to detect malicious automatically built domain names [10]. They used several methods and features to group the DNS queries. Then for each group, metrics such as the Jaccard index were computed to differentiate the domain names. J. Ma et al. employed supervised learning techniques (Naive-Bayes, SVM and Logistic Regression) to detect malicious websites from suspicious URLs [11]. To characterize the URLs, two categories of feature were used: lexical and host-based features. M. Antonakakis et al. presented a dynamic reputation system, Notos [12]. Using DNS query data and analyzing zone and network features of domains, Notos builds models of legitimate and malicious domain names.

In this work, our goal is to explore the application of evolutionary techniques in order to automatically generate signatures to detect botnets based on monitoring and analyzing domain names, specifically the ones that are used by botnets for domain fluxing. To the best of our knowledge, all of the works in the literature employ specific pre-defined features of domain names, however we aim to avoid this by only considering the domain names string sequence.

3 Methodology

Network security administrators proposed different approaches to deal with botnets that apply domain fluxing techniques. Some have used black lists to filter out the known C&C servers' domain names or pre-register the probable domains. Others used anomaly and signature-based detection methods. However, all require some type of knowledge on the Botnet domains or Domain Generation Algorithms (DGAs) to be able to generate the exact same domain lists as the botnets. This is a very costly (resources and time) process and also needs to be repeated each time a new DGA is injected. To this end, we believe that a light weight malicious domain name detector can go a long way. Thus, we propose a detection system just based on the automatic analysis of domain name records without requiring any a priori feature sets.

Indeed, one challenge is that automatically generated domain names are also used for legitimate background communications such as software updates and load balancing. Moreover, various well-known websites such as Google and Facebook also use this type of domains. Therefore, the first step to detect the botnet malicious domains is to differentiate legitimate automatically generated domain names from malicious ones. In this case, we propose a new SBB-based classifier system, Stateful-SBB, which works on domain name record strings using no a priori knowledge. We compare our proposed system against original SBB, C4.5, AdaBoost and Naive-Bayes based classifiers, where all require input based on a priori knowledge.

3.1 Learning Algorithms Employed

Naive-Bayes. A Naive-Bayes classifier is a simple probabilistic classifier based on the Bayes theorem, which assumes that the presence of an attribute in a given class is independent of other attributes. The classifier uses the method of maximum likelihood (probability) for parameter estimation. Given a training set (X,Y) where for each sample (x,y), x is an n-dimensional vector and y is the class label out of k number of classes, $C_1, C_2 \ldots C_k$, the classifier predicts that the sample belongs to the class C_i having the highest posteriori, conditioned on x ($P(C_i \mid x) > P(C_j \mid x)$ for $1 \leq j \leq k, j \neq i$). A more detailed explanation of the algorithm can be found in [13].

C4.5. C4.5 is a well-known decision tree-based learning algorithm, which uses the training data to create a tree structure and then classifies the new samples of the test data using the trained tree model. It employs a normalized information gain criterion to select attributes from a given set to determine the splitting point. In this process, the attribute with the highest information gain value is chosen to be the best point. A decision node is then generated based on the best point. The training process recursively continues on the sub-lists obtained until all of the data samples associated to the leaf nodes are of the same class or the classifier runs out of training samples. A more detailed explanation of the algorithm can be found in [13].

AdaBoost. Machine learning techniques' goal is to generate a rule that can predict the new test samples with a high accuracy. Creating a highly accurate rule is a difficult task but on the other hand, generating a set of rough rules of thumb with moderate

accuracy is not that hard. Based on this observation, boosting method starts with finding the rules of thumb called weak learner. Given the training set, AdaBoost calls the weak learning algorithm repeatedly, each time feeding it with a different distribution over the training data. Each call generates a weak classifier. At the end, the algorithm combines the classifiers to a single one that is much more accurate than any of them. A more detailed explanation of the algorithm can be found in [13].

SBB. SBB is a form of genetic programming based learning algorithm. It has a team-based framework in which a group of learners are employed to solve a problem. The algorithm consists of three populations: the point population, the team population and the learner (symbiont) population, Fig. 3. The learner population declares a set of symbionts whereas the team population declares learner teams and finally the point population denotes indexes to subsets of exemplars from the training data. Individuals in the learner population take the form of bid-based genetic programs or a representation consisting of program and (scalar) action. Thus, when evaluating a team, each learner program is executed, but only the learner with maximum output (bid) suggests its action (class label). The process repeats for each exemplar in the point population, and again for all teams. There are three important characteristics of learners in case of bidding. First, each learner bids on the point separately but only the learner action with the highest bid is returned as the team action. Second, using the data set with a fixed number of features for exemplars, the learners bid on each point based on the whole feature set. Finally, each learner resets its registers before bidding on the next point. A more detailed explanation of the algorithm can be found in [4].

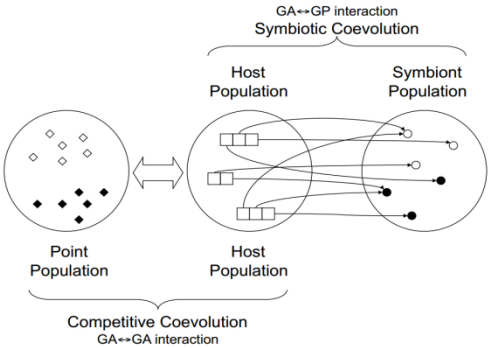


Fig. 1. SBB team-based mechanism [4]

Stateful-SBB. As discussed earlier, identifying the correct set of attributes, which properly represent the domain name characteristics, is challenging especially given that DGAs are moving targets. Thus, to this end, we explored using the raw domain names as an option for detection purposes. Thus, we designed and developed the Stateful- SBB to classify the malicious vs. non-malicious domains by only using the raw domain names. In other words, we explored how far we can push the classification performance without any a priori knowledge about the characteristics of the domain names, i.e. without any lexical features or packet level features.

Learners in the original SBB classifier bid based on all attributes of points (domain names) similar to the aforementioned classifiers. However, given a data set of variable length domains, neither original SBB nor the aforementioned classifiers can be used. Therefore, we change the SBB interface to bid based on each character of a domain name. The new model keeps the state information for each exemplar, hence we call it the Stateful-SBB. Figure 2. summarizes the team-learner interaction mechanism in the Stateful-SBB. Data set exemplars in the new layout are the variable length domain names. Features are the ASCII codes of the domain names' characters. A team receives a complete domain name but it passes the domain name to its team of learners character by character. Each learner then provides a bid per character as opposed to per exemplar. The learners' action that outbids the others is assigned as the team output for that specific character. Domain characters are related to each other and are not independent. To achieve the correlation of characters reflected in the bidding process, learners reset their registers only at the beginning of each specific domain name, not for every bid process on every character in a domain. At the end of each domain name (when all the learners bid on the entire domain name characters), a team will have a sequence of the best learners' actions as the team output sequence. Finally, the team will decide on its final action for that specific domain name. Different policies can be used for the final action selection of a team, which is discussed in the evaluation section.

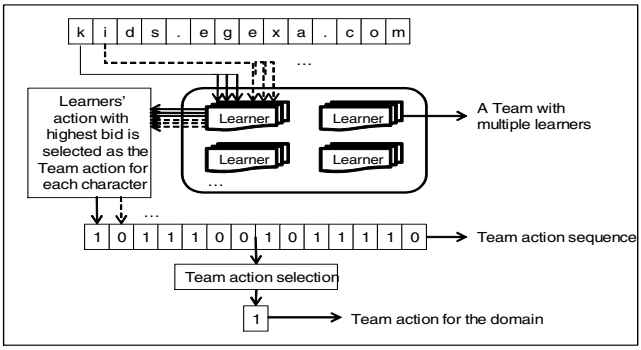


Fig. 2. Stateful-SBB mechanism

4 Evaluations and Results

In this work, the data set employed is collected from various resources including the publicly available botnet C&C domain lists such as Amada [14] and ZeuS [15]. Additionally, most frequently requested domains from the Alexa list [16] are used as the legitimate domain names. These include known C&C activity as well as social network sites such as Facebook backend and antivirus upload. Thus, the string format resulting data set goes beyond just the C&C traffic and includes other legitimate traffic observable at DNS level. Table 1 details the DNS data set employed in this work. "Class 0" represents normal automatically generated domain names and "class 1" represents the malicious automatically generated ones.

Other than the Stateful-SBB classifier, other classifiers require the data set to be feature-based (a priori knowledge) for training and testing. Thus, we employed a heuristics-based feature extraction on the components of a given domain name to compare against our proposed system. Each domain name has three components: (i) top level domain (TLD), (ii) core domain, and (iii) sub-domain. For example, in mail.google.com, com is the TLD, google is the core domain, and mail is the sub-domain. Given that the TLD names are distinctive and fixed, we only use the other two components (core domain and sub-domain) in feature extraction. Thus, in this work, for each domain name, a set of 17 features are extracted, Table 2. The first 14 features are based on the sub-domain and the last 3 are based on the core-domain component. Overall, the features aim to highlight the structural anomalies in the domain names (as seen in the literature). In other words, domain names that are not likely to be typed by a person. To detach the top-level domain and to extract the feature #12, we employ the Mozilla suffix list [17].

Table 1. Summary of the DNS data set employed

Data set	Data set total Num. of Samples			206
	"Class 0" total Num. of Samples			123
	"Class 1" total Num. of Samples			163
	Training		**Testing**	
	Class0	**Class1**	**Class0**	**Class1**
	90	116	33	47

Table 2. Feature set definition

No.	Features
1	Domain starts with "www"
2	Total sub-domain length: number of characters in all sub-domains (minus the dots)
3	Number of sub-domains: number of sub-domain blocks.
4	Maximum sub-domain length: the largest sub-domain block length
5	10+ character sub-domain count: number of sub-domains longer than 10 chars
6	1 character sub-domain count: number of sub-domains with one char
7	Contains IP: A Boolean flag. If there exists four sub-domain blocks between 0-255, following each other.
8	Alphabetic ratio: Num. of alphabetic character in all sub-domains divided by character count
9	Hexadecimal ratio: Num. of hexadecimal digits (A-F,a-f, 0-9) divided by character count
10	Standard deviation of sub-domain lengths
11	Non-alphanumeric ratio: Number of non-alphanumeric characters
12	Contains imbedded TLD, if the sub-domains contain any items in the Mozilla suffix list
13	Contains imbedded file extension
14	Number of alphabetic to non-alphabetic and vs. transitions
15	Core-domain length
16	Core-domain alphabetic character ratio
17	Core-domain alpha to non-alpha and vs. transition count

We trained each classifier (NB, C4.5, AdaBoost, SBB and Stateful-SBB) on the training data set to identify maliciously generated domain names from the non-malicious ones. Then, the trained models are tested on the test data set. To this end, we divided the dataset into two parts (training and testing) based on: (i) An almost 30-70% breakdown for test and training, respectively; and (ii) keeping enough samples of each class in both of the data sets. It should be noted here that default parameters in WEKA [18] are used for Naive Bayesian, AdaBoost and C4.5 (pruned) classifiers, whereas parameters given in [19] are used for both the original and the Stateful-SBB.

Table 3 presents the results of these experiments. As the results show, the best performers are the C4.5 classifier and the Stateful-SBB classifier. These results show that it is possible to identify the maliciously generated domain names with a high detection rate and a low false positive rate even without a priori knowledge, i.e. without a specific feature set extracted from lexical attributes of a domain name.

It should be noted here that we employed exactly the same data set for all the classifiers. The only difference is for the classifiers other than Stateful-SBB, we represented each record of the data set using the 17 features given in Table 2. However, in the case of Stateful-SBB, we represented the data set in its ASCII code to the classifier. As discussed earlier, the team final action of the Stateful-SBB should be chosen from its learners' output sequence, which is constructed during the bidding process using the domain characters. Given that the domain names are a composition of related characters in a meaningful order, there are some important questions that need to be answered: Is it necessary to use all the domain name characters in the learning process to have a relatively good output label? Should the combination of all actions in the sequence be considered or just the last one? We run several experiments (7 different approaches to the Stateful-SBB and 20 runs for each approach) and evaluated the proposed system on different action selection methods to answer these questions. Because of page limitations, we are not able to present all of these experiments. However, our experimental results showed that the best performances were achieved by the “Last-best” and the “Most-freq” team action selection methods. The "Last-best" method assumes that the class label for the domain name is that returned at the last character. As the learners would not reset their registers in the bidding process of a domain, the last action of the sequence is somehow affected by all the actions in the sequence, where all the domain characters are considered. As the "Last-best" heuristic team might not always reflect all the best actions of bidding process, "Most-freq" method chooses the most frequent action of a team action sequence to be the team final decision. So these versions of the Stateful-SBB are chosen for the evaluation of the proposed system against the other classifiers. Given the results of the Stateful-SBB with two proposed settings, it can be concluded that our proposed Stateful-SBB using the “Last-best” action selection procedure is slightly better than the others. The Results are shown in Table 3.

However to be more precise, we run a T-Test (between the pruned C4.5 classifier and the Stateful-SBB) on 20 different experiments of each solution. The T-Test results indicate that there is not a statistically significant difference between the pruned C4.5 based classifier, which requires a priori known feature set, and the Stateful-SBB, which does not require such a priori information. In summary, these results show that the proposed system employing the new Stateful-SBB has a high

classification rate with zero false negative rate for "class 0" (non-malicious domain names). Given that misclassifying legitimate domain names as malicious ones (which ends in a blocking action) can interfere with a genuine website activity, the performance of Stateful-SBB is very promising and shows that it could be deployed in real world environments. Moreover, the most critical phase for all the classifiers using a set of pre-defined features (Naive-Bayes, AdaBoost, C4.5 and SBB) is the feature extraction. In this phase, identifying a correct set of features (a priori information) that can properly represent the domain names is crucial. However, even if a valuable feature set can be identified, since botmasters change their DGAs frequently to evade the detection systems, a feature set that works before, could be stale when the DGA changes! On the other hand, the Stateful-SBB can be applied directly to the domain name strings (using their ASCII codes only) without requiring any feature extraction, in other words without any a priori knowledge. Having said this, the Stateful-SBB training time (computation time) is longer than the others. However, because the training can be performed offline, we believe that the benefits of the proposed system can be considered as a major improvement in this field.

Table 3. Evaluation results on the test data set

		Naive Bayes	Pruned C4.5	AdaBoost	SBB	"Most-freq" Stateful-SBB	"Last-best" Stateful-SBB
Classification Rate		0.95	0.96	0.95	0.95	0.98	0.99
Class 0	Precision	0.94	0.92	0.91	0.94	1	1
	Recall	0.94	1	0.97	0.97	0.94	0.97
	F-Measure	0.94	0.96	0.94	0.95	0.94	0.97
Class 1	Precision	0.96	1	0.98	0.97	0.94	0.97
	Recall	0.96	0.94	0.97	0.94	1	1
	F-Measure	0.96	0.97	0.97	0.95	0.97	0.99
Training Time (sec)		0.01	0.03	0.07	121.59	4336.37	3763.62

Table 4. T-Test results

	SBB “Last-Best”	SBB “Most-Frequent”
Pruned C4.5	0.11	0.50

5 Conclusion and Future Work

Legitimate users are not the only ones that use DNS to communicate. Modern botnets avoid ‘hardcoding’ the address of the C&C server because if the C&C server is identified, they can be blocked at the firewall level. DNS provides a scalable solution for botnets since a list of domain names can be passed to the victim host as C&C server. As long as the victim manages to connect to the server using one of the domains in the list, it will download the malware and join the botnet. On the other

hand, all the domains on the list (whether they resolve to an IP address or not) need to be blacklisted to be able to fully mitigate the attack.

Fortunately for the defenders, DNS traffic of a botnet exhibits abnormal properties that can be detected. The most important property is the structure of the domains that are being queried, i.e. long, with many sub-domains and seemingly random set of characters. Thus, a suitable solution is to monitor the communications at the DNS level to detect abnormal query patterns, specifically queries that a human would not possibly be able to type, based on temporal, structural and syntactic properties.

In this work an evolutionary computation based solution is investigated. To this end, we designed and developed the Stateful-SBB classifier, which is utilized to support variable length input. In addition to providing a very high accuracy on classification and automatically generating signatures, Stateful-SBB identifies the set of attributes to be used in classification automatically without requiring any a priori knowledge, whereas typical classifiers evaluated requires a fixed set of features extracted based on a priori knowledge. The results show that Stateful-SBB based system performs comparable to other classification methods without requiring a feature set to be determined a priori. Future work will include improvement on the training time of the Stateful-SBB, evaluating other classifiers such as SVM and testing the proposed system under other data sets.

Acknowledgments. This research is supported by the Natural Science and Engineering Research Council of Canada (NSERC) grant, and is conducted as part of the Dalhousie NIMS Lab at http://projects.cs.dal.ca/projectx/.

References

1. Feily, M., Shahrestani, A.: A Survey of Botnet and Botnet Detection. Emerging Security Information. In: Emerging Security, Systems and Technologies (2009)
2. Damballa Inc.:Top 10 Botnet Threats (2010), `http://www.damballa.com`
3. McAfee Labs Thread Reports, `http://www.mcafee.com/apps/view-all/publications.aspx`
4. Doucette, J., McIntyre, A.R., Lichodzijewski, P., Heywood, M.I.: Symbiotic Coevolutionary Genetic Programming: A Benchmarking Study Under Large Attribute Spaces. Genetic Programming and Evolvable Machines 13(1), 71–101 (2012)
5. Vuong, S.T., Alam, M.S.: Advanced Methods for Botnet Intrusion Detection Systems. In: Intrusion Detection Systems. InTech. (2011)
6. Bailey, M., Cooke, E., Jahanian, F., Xu, Y., Karir, M.: A Survey of Botnet Technology and Defense. In: CATCH 2009 (2009)
7. The Role of DNS in Botnet Command & Control. In: Open DNS Inc., Whitepaper (2012)
8. Zhang, L., Yu, S., Wu, D., Watters, P.: A Survey on Latest Botnet Attack and Defence. In: TrustCom, pp. 53–60 (2001)
9. Stalmans, E., Irwin, B.: A Framework for DNS Based Detection and Mitigation of Malware Infections on a Network. In: Information Security South Africa (2011)
10. Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S.: Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis. IEEE/ACM Transaction on Networking 20, 1663–1977 (2012)

11. Ma, J., Saul, L.K., Savage, S., Voelker, G.: Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs. In: ACM KDD (2009)
12. Antonakakakis, M., Perdisci, R., Dagon, D.: Building a Dynamic Reputation System for DNS. In: USENIX Security 2010 (2010)
13. Alpaydin, E.: Introduction to Machine Learning. MIT Press (2004)
14. Abuse: AMaDA, https://palevotracker.abuse.ch/
15. Abuse: Zeus Tracker, https://zeustracker.abuse.ch/
16. Alexa, http://www.alexa.com/topsites
17. Top Level Domain Names, http://mxr.mozilla.org/mozilla-central/source/netwerk/dns/effective_tld_names.dat?raw=1
18. WEKA, http://www.cs.waikato.ac.nz/ml/weka/
19. Lichodzikewski, P., Heywood, M.I.: Symbiosis Complexification and Simplicity under GP. In: GECCO 2010 (2010)

Evolving Gaits for Physical Robots with the HyperNEAT Generative Encoding: The Benefits of Simulation

Suchan Lee[1], Jason Yosinski[1], Kyrre Glette[2], Hod Lipson[1], and Jeff Clune[1,3]

[1] Cornell University, USA
[2] University of Oslo, Norway
[3] University of Wyoming, USA
{sl746,jy495,hod.lipson,jeffclune}@cornell.edu, kyrrehg@ifi.uio.no

Abstract. Creating gaits for physical robots is a longstanding and open challenge. Recently, the HyperNEAT generative encoding was shown to automatically discover a variety of gait regularities, producing fast, coordinated gaits, but only for simulated robots. A follow-up study found that HyperNEAT did not produce impressive gaits when they were evolved directly on a physical robot. A simpler encoding hand-tuned to produce regular gaits was tried on the same robot, and outperformed HyperNEAT, but these gaits were first evolved in simulation before being transferred to the robot. In this paper, we tested the hypothesis that the beneficial properties of HyperNEAT would outperform the simpler encoding if HyperNEAT gaits are first evolved in simulation before being transferred to reality. That hypothesis was confirmed, resulting in the fastest gaits yet observed for this robot, including those produced by nine different algorithms from three previous papers describing gait-generating techniques for this robot. This result is important because it confirms that the early promise shown by generative encodings, specifically HyperNEAT, are not limited to simulation, but work on challenging real-world engineering challenges such as evolving gaits for real robots.

1 Introduction

Legged robots can operate in a much wider range of environments than their wheeled counterparts. However, designing gaits for legged robots is a difficult and time-consuming process for human engineers [16, 24], and must be repeated every time a robot is created or modified [11]. Scientists thus investigate how to automatically produce gaits via machine learning and evolutionary algorithms, and the result is often a better gait than those created by human engineers [10–12, 23, 25]. While it has been shown that gaits perform better if they are *regular*—i.e., that they have coordinated movements, such as left-right symmetry or front-back symmetry [4, 6, 7, 23]—experimenters usually have to explicitly decide and specify these regularities [1, 15, 23, 22]. Such manual intervention is time consuming, requires expert knowledge, and adds constraints that may hurt performance.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 540–549, 2013.
© Springer-Verlag Berlin Heidelberg 2013

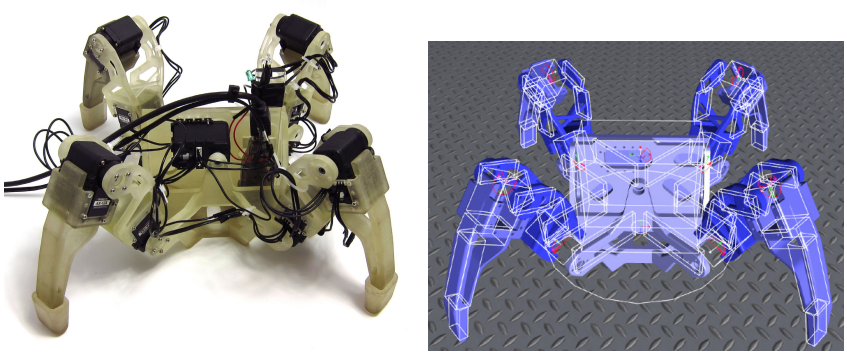


Fig. 1. The QuadraTot robot platform on which gaits were evolved. **Left:** The physical robot, which is composed of 3-D printable and off-the-shelf components. **Right:** The representation of the robot in the simulator.

Previous work has shown that the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) generative encoding [20] can automatically generate a variety of regular gaits that outperform gaits evolved with direct encodings [4, 7]. However, that work only verified these claims in simulation. Yosinski et al. evolved gaits with HyperNEAT directly in hardware on the QuadraTot robot platform (Figure 1). They found that HyperNEAT's gaits outperformed manually designed, parameterized learning algorithms, but still did not produce impressive, natural gaits [25].[1] A follow-up study built a simulator for QuadraTot to test whether the inclusion of a simulator would improve results and found that it did: when gaits were evolved in simulation with a simple direct encoding and then transferred to the QuadraTot robot, the resulting gaits were faster than the gaits produced by evolving gaits with HyperNEAT directly on the robot [9]. The simulator helped because it afforded much larger population sizes and more generations than were possible when evolving directly in hardware, resulting in 333 times more evaluations per run (60000 vs. 180) [9, 25].

The work with the simulator [9] evolved gaits with a simple encoding manually constrained to produce specific regularities. The success of that work raises the question of whether the performance gains were due to the added benefit of a simulator or the use of a simple, hand-designed encoding. We hypothesized that HyperNEAT, which has been previously shown to automatically discover complex regularities to produce high-performing gaits [7, 25], would outperform the simpler encoding from Glette et al. if combined with a simulator. Here we test that hypothesis by evolving gaits with HyperNEAT in the same simulator from Glette et al. and then transferring those gaits to the QuadraTot robot. Our experiments confirmed the hypothesis: with the simulator, HyperNEAT evolved the highest-performing gaits observed to date for the QuadraTot platform.

[1] Videos available at `http://creativemachines.cornell.edu/evolved-quadruped-gaits`

2 Methods

Robot Hardware. We performed experiments on the QuadraTot quadrupedal robot platform (Figure 1-Left) [25]. It has 9 degrees of freedom: two joints per leg and one joint that rotates along the robot's midline. The QuadraTot hardware designs and the software for this project are open source[2], and all hardware components are either off-the-shelf or 3D-printed. There are results on the platform for nine different learning algorithms from three previous publications [9, 18, 25].

The joints are powered by Robotis Dynamixel servos; five AX-18A servos for the inner joints of each leg and the single midline joint, and four AX-12A servos for the the outer joints of each leg, which require less power and can thus have less expensive motors. Each servo has a built-in safety mechanism that shuts itself off to prevent damage if the servo's current, range, temperature, or torque is too high. This safety mechanism activated frequently and inconsistently, adding significant noise to the evaluation process. As pointed out in a previous study [25], evolved gaits on QuadraTot are highly variable and produce many shutdowns because they force the servos to exert too much torque. To prevent collisions between different pieces of the robot's body, we limited the allowable range of movement for the inner, outer, and hip joints to [-85°, +60°], [-113°, +39°], and [-28°, +28°], respectively. We also implemented the Smart Cropping System from Shen et al. [18], which prevents combinations of joint positions for the inner and outer joint of each leg that generate extreme amounts of torque. A final method of reducing torque was to reduce the weight of the robot. Yosinski et al. had the small Linux computer that performed all computation on the robot, but we removed it and sent commands from it to the robot via a cable. We tracked the robot's position using an infrared LED observed by a Wiimote.

Simulator. Gaits evolved in the simulator from Glette et al.[2], which represents QuadraTot in the Nvidia PhysX physics engine, including the mass and size of the QuadraTot components and its degrees of freedom (Figure 1-Right). In the simulator, each individual joint range was limit as described above, but Smart Cropping was not included because we found that it hindered performance by limiting the types of gaits evolution could explore in early generations.

HyperNEAT. HyperNEAT is an algorithm for evolving artificial neural networks (ANNs) [20]. It has been repeatedly described in detail [7, 8, 20], so here we provide only a summary. Instead of directly encoding each ANN weight individually on the genome, in HyperNEAT the genome is a compositional pattern producing network (CPPN) [19]. The CPPN specifies the weights in a similar way to how natural organisms develop. In nature, phenotypic attributes are specified as a function of their geometric location, and such positional information is conveyed through chemical morphogen gradients [3]. For example, the concentration of one chemical could indicate the position along the head-to-tail axis and

[2] A parts list, hardware CAD files, software (including the simulator), and gait videos are available at `http://creativemachines.cornell.edu/evolved-quadruped-gaits`

another chemical in bands could indicate if a cell is in an odd- or even-numbered segment. Based on the relative concentrations of these chemicals, a cell can know where it is geometrically and, thus, what type of cell to become [3].

With CPPNs this process is abstracted as a network of math functions that operate in a Cartesian geometric space. The coordinates of phenotypic elements are provided as inputs to the CPPN and the outputs specify phenotypic traits. For example, when CPPNs encode 2D pictures, the coordinates of each pixel are iteratively input into the genome and the output is the grayscale value at that coordinate [17]. Because a CPPN network is composed of math functions, these functions can create geometric regularities in the phenotype. For example, a Gaussian function of an axis can provide symmetry (e.g. left-right), and a repeating function (e.g. sine) of an axis could provide repetition (e.g segmentation). Both 2D pictures and 3D objects evolved with CPPNs look like natural and engineered objects, and contain complex regularities, such as symmetries and repeated motifs, with and without variation [5, 17].

In HyperNEAT, CPPNs encode the weights of the connections between neurons as a function of the geometric locations of those neurons (Figure 2). The Cartesian coordinates of the two neurons at the end of each connection are input into the CPPN, and the output is the weight of that connection. If the output is smaller than a threshold, the weight is set to zero, functionally removing the connection. The process is repeated for each possible connection. Just as in 2D pictures and 3D objects, the CPPN can create complex, regular, geometric patterns (e.g. left-right symmetry or repeated modules), but in this case the patterns are in the weights of a neural network [7]. The neural regularities produced by HyperNEAT enable significantly improved performance on problems that are regular [7, 20], including evolving quadruped gaits in simulation [4, 6, 7].

In HyperNEAT, the CPPN genomes evolve via the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [21], which has three major components. First, NEAT starts with small genomes that encode simple networks and complexifies them via mutations that add nodes and links to the networks. Second, NEAT has a fitness-sharing system that preserves diversity and allows for new innovations to be tuned by evolution before competing them against more adapted rivals. Third, historical information is recorded that facilitates crossover in a way that is effective, yet avoids the need for expensive topological analysis. A full explanation of NEAT can be found in Stanley and Miikkulainen 2002 [21].

In this study, the ANN inputs, outputs, activation functions, and the size of the hidden layer are the same as in Yosinski et al. [25]. The ANN had a fixed topology of three 3×4 Cartesian grids of nodes for the input, hidden, and output layers. The inputs to the substrate were the angles requested in the previous time step for each of the 9 joints of the robot and a sine and cosine wave to facilitate periodic motion. The outputs of the substrate at each time step were nine numbers (for each joint) in the range [-1, 1] which were scaled to the allowable ranges for the servos. As in Yosinski et al. [25], we generated pseudo-positions at 160Hz and then downsampled over consecutive blocks of four time steps to obtain the actual commanded positions at 40Hz; this reduced the

number of gaits which commanded switches from extreme negative to extreme positive numbers at 40Hz, which overly taxed the servos.

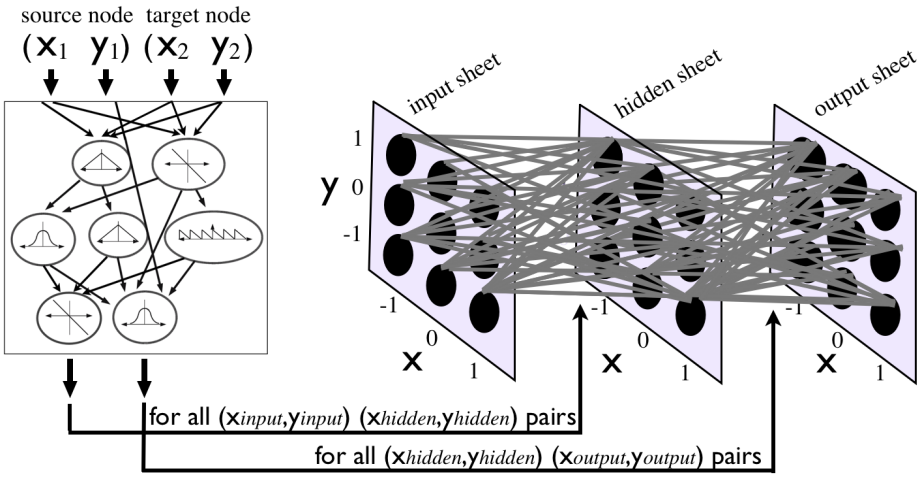


Fig. 2. A CPPN specifying a neural network. In HyperNEAT, weights are a function of the Cartesian coordinates of the source and target node for each connection. All pairwise combinations of source and target node coordinates are iteratively passed into a CPPN to determine the weight of each ANN link. Figure from Clune et al. [7].

Evolutionary Process and Parameters: Each run had a population size of 200 and lasted 200 generations. We performed 20 HyperNEAT runs that differed only in the seed provided to the random number generator, which affected stochastic events such as mutation. To make a statistical comparison to the encoding from Glette et al., we conducted 19 runs using the original code to supplement the one run performed for that paper. Each gait was evaluated for fourteen seconds in reality, with interpolation from and to a stationary position in the first and last two seconds, respectively, as in Yosinski et al., effectively resulting in 12 seconds of full-speed motion. In simulation gaits were evaluated for 12 seconds. All HyperNEAT parameters were identical to those in Yosinski et al. except for the frequency of the sine wave input to the ANN, which was lowered from 4.2Hz to 0.64Hz to reduce servo shutdowns. To further reduce servo shutdowns, we punished high-frequency gaits during evolution. We calculated the frequency of a gait as the average number of servo direction changes per leg per second. If this frequency was higher than the experimentally-selected threshold of 1.67 Hz, the measure of distance traveled by the center of mass during the gait was reduced exponentially by multiplying it by a discount factor of $e^{(1.67-\text{freq})}$. Following Clune et al. [4], the fitness equation was 2^{distance^2}. After evolving the gaits in simulation, the champion gait of the last generation of each of the 20 runs was transferred onto the real robot and the distance traveled was measured.

3 Results and Discussion

The simulator enabled HyperNEAT to evolve fast, natural gaits. In simulation, HyperNEAT gaits were faster than those from Glette et al. (Figure 3-Top, $p < 6.8 \times 10^{-8}$ when comparing the best gaits in the final generation of each run via Matlab's Wilcoxon rank sum test). Specifically, HyperNEAT gaits were 52.1% faster in simulation (25.4 cm/s $\pm$ 3.4 SD versus 16.7 cm/s $\pm$ 1.9 SD). To facilitate comparisons to earlier works [9, 25] we report mean $\pm$ SD, but our qualitative conclusions are the same when using medians. Plots of servo positions over time reveal that the evolved HyperNEAT gaits are regular and coordinated (Figure 3-Left), confirming previous results with HyperNEAT in simulation [4, 7].

On the physical robot, HyperNEAT gaits from this study outperformed gaits from all previous QuadraTot studies (Table 1) [9, 18, 25], including those of Glette et al. However, comparing performance in hardware between studies performed in different laboratories is difficult, not only because reality is inherently noisy, but because even two copies of the same robot are not identical and may produce different speeds for the same input gait. Gaits for this study and two previous studies [18, 25] were evaluated on a copy of the QuadraTot robot in the Cornell Creative Machines Lab (CCML), while the gaits in Glette et al. were evaluated on a different copy of the same robot in the Robotics and Intelligent Systems (ROBIN) lab at the University of Oslo. The two robots were evaluated on two different surfaces, and had different material enveloping their feet to increase friction (compare Figure 1-Left to Figure 2 of Glette et al).

The previous fastest gait on any copy of a QuadraTot was from Glette et al., and traveled 17.8 cm/s on the ROBIN QuadraTot. The fastest gait produced by HyperNEAT with a simulator in the experiments for this paper traveled 14.5 cm/s on the CCML QuadraTot. It was unclear whether this difference in performance was due to the differences in the gaits themselves or dissimilarities between hardware. To control for this possibility, we ran the fastest gait from Glette et al. 10 times on CCML and measured a mean speed of only 12.95 cm/s $\pm$0.93 SD (vs. 17.8 cm/s measured on ROBIN) and a maximum of 13.8 cm/s (Table 1). It thus appears that the CCML version of the robot is slower. Moreover, the HyperNEAT gait (14.5 cm/s) is faster than the best gait of Glette et al. on the CCML QuadraTot. Unfortunately, it was not possible to test the best HyperNEAT gait on the ROBIN QuadraTot. Because HyperNEAT outperformed the simple encoding from Glette et al. on the same robotic hardware, we tentatively conclude that HyperNEAT produces faster gaits for the QuadraTot robot. This conclusion is supported by the fact that HyperNEAT also outperformed the encoding from Glette et al. in simulation.

We now discuss the differences between the gaits evolved by HyperNEAT directly on the physical robot [25], and those first evolved in a simulator and then transferred to the robot (this study). On the physical robot, the gaits produced by HyperNEAT with a simulator were faster, more natural, and more repeatable than those evolved directly on the QuadraTot robot [25]. The gaits were also more regular, as they were in simulation (Figure 3-Left). This result is important because it confirms that HyperNEAT can produce the important property of

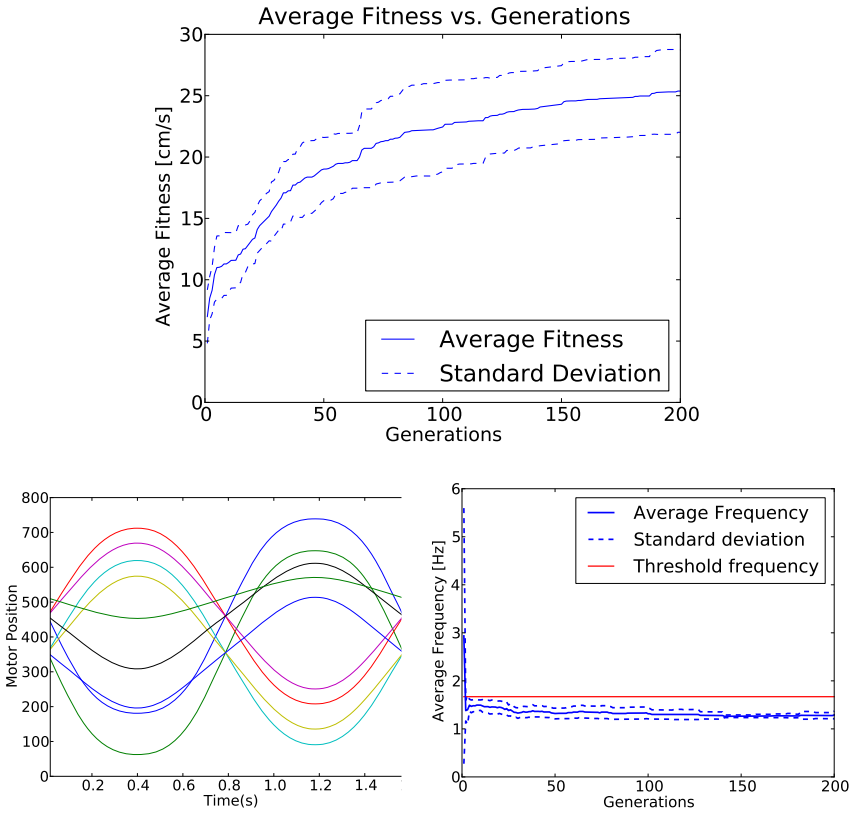


Fig. 3. Top: HyperNEAT outperforms a genetic algorithm with a simple encoding [9] when both algorithms are combined with a simulator. Plotted are means over 20 runs in simulation (solid lines) ± SD (dashed lines). HyperNEAT gaits are 52.1% faster in simulation and 5.1% faster in reality than those from a previous study [9] (details in Table 1). **Left:** Servo positions over time (for nine servos) for a representative simulated HyperNEAT gait. HyperNEAT produced smooth and symmetrical gaits that contained complex regularities. **Right:** Mean gait frequency averaged over 20 runs. Gaits with frequencies above a threshold (horizontal line) receive a fitness penalty. HyperNEAT quickly learned to produce gaits with frequencies low enough to avoid this penalty.

regularity in a challenging, real-world domain, which was not previously observed when evolving directly on the hardware [25]. Producing regular solutions is a key to exploiting regularity in difficult engineering problems [7].

The simulator likely improved performance because of the number of evaluations it enabled, both in terms of the population size (200 vs. 9) and the number of generations (200 vs. 20), leading to a total difference of 40000 vs. 180 per evolutionary run compared to Yosinski et al. [25]. Another potential cause of improved performance is the lower noise in the simulator, which could have helped HyperNEAT find coordinated, regular, gaits, which perform better. On the physical

Table 1. Velocities of evolved gaits in simulation and on two different copies of the QuadraTot robot. Subject to availability, data are reported from the experiments for this paper and three previous studies. Reported are the total number of evaluations per run, the mean of the fastest gaits produced in each run in simulation, and the single fastest gait produced on the CCML and ROBIN copies of the QuadraTot robot (see text for their differences). *Instead of using the single fitness value reported in [9], we ran 19 additional runs and used the mean fitness of those 20 runs. Velocities are in cm/s, and bold indicates the best performance. **The median fitness that corresponds with this mean is 26.9 cm/s, 95% confidence interval [23.8 cm/s, 26.75 cm/s].

	Evaluations	Simulated Velocity	Real Vel. (CCML)	Real Vel. (ROBIN)
Parameterized gaits + optimization [25]	153	–	5.8	–
HyperNEAT in hardware [25]	180	–	9.7	–
RL PoWER Spline [18]	300	–	11.1	–
GA + simulator [9]	60000	*16.7	13.8	**17.8**
HyperNEAT + simulator [this paper]	40000	****25.4**	**14.5**	–

robot, the noise in the evaluation was substantial, preventing effective learning [25]. To investigate this hypothesis, we performed 20 runs in simulation with only 180 fitness evaluations, which was the number used in Yosinski et al. [25]. The simulated gaits performed slightly, but not significantly, better than those evolved in hardware ($p = 0.1571$, mean fitness 7.9 ± 2.14 cm/s). Reduced noise thus may have had a small affect on performance, but the substantial performance gains that resulted from using a simulator likely came from the additional evaluations the simulator afforded. The encouragement of low-frequency gaits in this study also may have aided performance, especially since in simulation the gaits were high-frequency in a few early generations before rapidly settling to a range below the penalized threshold (Figure 3-Right).

While this study was able to produce the fastest QuadraTot gait to date, most of the gaits in simulation did not transfer well to reality. Many gaits that were fast in simulation performed poorly on the real robot, largely due to servos that were too weak and shut down, or because of differences between simulation and reality. Repeated attempts to minimize these problems were unsuccessful. In future studies we will use a robot that has more mechanical advantage and requires less torque from each servo, such as the Aracna platform [14]. That we did not model the servos in simulation, especially with their frequent failures, suggests that even better results could be obtained via a simulator that contained or learned servo models. In future work we will also incorporate techniques to minimize the gap between the simulator and reality [2, 13, 26].

4 Conclusion

With a simulator, HyperNEAT evolved the fasted gait yet recorded for the QuadraTot robot, outperforming nine machine learning algorithms from three

previous publications [9, 18, 25], including an improvement of 52.1% in simulation and 5.1% in reality over the previous best QuadraTot gait by Glette et al. These results provide an important demonstration that the HyperNEAT generative encoding can evolve state-of-the-art results for challenging engineering problems, in this case evolving gaits for a legged robot. The results further reaffirm the benefits of using simulators when solving real-world challenges with evolutionary algorithms. Our results additionally confirm—with a different robot and simulator—previous work that has shown that HyperNEAT is an effective encoding for automatically evolving coordinated, regular gaits in simulation [4, 7, 25]. That HyperNEAT outperformed the encoding hand-designed by Glette et al. shows that HyperNEAT can outperform even evolutionary algorithms manually designed to incorporate human domain knowledge regarding which regularities are thought to be beneficial for a problem. The results thus demonstrate that automatically discovering regularities can be a superior approach to specifying them, even on problems that are relatively well-understood.

References

1. Beer, R., Gallagher, J.: Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior 1(1), 91–122 (1992)
2. Bongard, J.C.: Synthesizing Physically-Realistic Environmental Models from Robot Exploration. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 806–815. Springer, Heidelberg (2007)
3. Carroll, S.: Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom. Norton, New York (2005)
4. Clune, J., Beckmann, B., Ofria, C., Pennock, R.: Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2764–2771 (2009)
5. Clune, J., Lipson, H.: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In: Proceedings of the European Conference on Artificial Life, pp. 144–148 (2011)
6. Clune, J., Ofria, C., Pennock, R.: The sensitivity of HyperNEAT to different geoemtric representations of a problem. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 2764–2771 (2009)
7. Clune, J., Stanley, K.O., Pennock, R., Ofria, C.: On the performance of indirect encoding across the contiuum of regularity. IEEE Transactions on Evolutioanry Computation 15, 346–367 (2011)
8. Gauci, J., Stanley, K.: Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 997–1004. ACM (2007)
9. Glette, K., Klaus, G., Zagal, J., Torresen, J.: Evolution of locomotion in a simulated quadruped robot and transferral to reality. In: Proceedings of the Seventeenth International Symposium on Artificial Life and Robotics (2012)
10. Hornby, G., Lipson, H., Pollack, J.B.: Generative representations for the automated design of modular physical robots. IEEE Transactions on Robotics and Automation 19, 703–719 (2003)

11. Hornby, G., Takamura, S., Tamamoto, T., Fujita, M.: Autonomous evolution of dynamic gaits with two quadruped robots. IEEE Transactions on Robotics 21(3), 402–410 (2005)
12. Kohl, N., Stone, P.: Machine learning for fast quadrupedal motion. In: The Nineteenth National Conference on Articifial Intelligence (AAAI), pp. 611–616 (2004)
13. Koos, S., Mouret, J., Doncieux, S.: The transferability approach: Crossing the reality gap in evolutionary robotics. IEEE Trans. Evolutionary Computation 1, 1–25 (2012)
14. Lohmann, S., Yosinksi, J., Gold, E., Clune, J., Blum, J., Lipson, H.: Aracna: An open-source quadruped platform for evolutionary robotics. In: Proceedings of the 13th International Conference on the Synthesis and Simulation of Living Systems, pp. 387–392 (2012)
15. Raibert, M., Chepponis, M., Brown Jr., H.: Running on four legs as though they were one. IEEE Journal of Robotics and Automation 2(2), 70–82 (1986)
16. Ridderstrom, C.: Legged locomotion control–a literature survey. In: Tech Report: Royal Institute of Technology. pp. 1400–1179. No. TRITA-MMK, Stockholm, Sweden (1999)
17. Secretan, J., Beato, N., D'Ambrosio, D., Rodriguez, A., Campbell, A., Folsom-Kovarik, J., Stanley, K.: Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. Evolutionary Computation 19(3), 373–403 (2011)
18. Shen, H., Yosinski, J., Kormushev, P., Caldwell, D.G., Lipson, H.: Learning fast quadruped robot gaits with the RL power spline parameterization. In: AIMSA Workshop on Advances in Robot Learning and Human-Robot Interaction (2012)
19. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Matter 8(2), 131–152 (2007)
20. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial Life 15(2), 185–212 (2009)
21. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
22. Téllez, R.A., Angulo, C., Pardo, D.E.: Evolving the Walking Behaviour of a 12 DOF Quadruped Using a Distributed Neural Architecture. In: Ijspeert, A.J., Masuzawa, T., Kusumoto, S. (eds.) BioADIT 2006. LNCS, vol. 3853, pp. 5–19. Springer, Heidelberg (2006)
23. Valsalam, V., Miikkulainen, R.: Modular neuroevolution for multilegged locomotion. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 265–272 (2008)
24. Wettergreen, D., Thorpe, C.: Gait generation for legged robots. In: IEEE International Conference on Intelligent Robots and Systems, pp. 1413–1420 (1992)
25. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., Lipson, H.: Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization. In: Proceedings of the 20th European Conference on Artificial Life, pp. 11–18 (2011)
26. Zagal, J., Ruiz-del-Solar, J., Vallejos, P.: Back to reality: Crossing the reality gap in evolutionary robotics. In: Proceedings of IAV 2004, the 5th IFAC Symposium on Intelligent Autonomous Vehicles (2004)

Co-evolutionary Approach to Design of Robotic Gait

Jan Černý and Jiří Kubalík

Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University, Technická 2, 166 27 Prague 6, Czech Republic
cernyj31@fel.cvut.cz, kubalik@labe.fel.cvut.cz

Abstract. Manual design of motion patterns for legged robots is difficult task often with suboptimal results. To automate this process variety of approaches have been tried including various evolutionary algorithms. In this work we present an algorithm capable of generating viable motion patterns for multi-legged robots. This algorithm consists of two evolutionary algorithms working in co-evolution. The GP is evolving motion of a single leg while the GA deploys the motion to all legs of the robot. Proof-of-concept experiments show that the co-evolutionary approach delivers significantly better results than those evolved for the same robot with simple genetic programming algorithm alone.

1 Introduction

Walking robots have advantage over wheeled ones when navigating complex and uneven environments. However, to fully use abilities of a particular legged robot it is necessary to optimize its gait specifically for its mechanical structure, dimensions and parameters.

This is a highly challenging task seeking for a coordinated control of many joints, given multiple (often contradictory) optimization objectives such as the maximal or some specific speed of locomotion, low-energy operational mode, stability of the robot, etc. The solution sought must also comply with multiple constraints. Typically, the state transitions have to be continuous in order to attain smooth gait patterns. Other constraints can be determined by limited resources and mechanical parameters that determine inherent capability limits of the robot [3]. Moreover, the optimization objectives as well as the constraints are non-linear, in general.

Obviously, manual generation of gaits is very difficult, if feasible at all, due to the aspects mentioned above. Utilization of standard numerical optimization methods is limited since the objective functions are not defined analytically (each candidate gait is evaluated by a real experiment or through a simulation), hence no information about continuity or differentiability is available. Single-state heuristic methods that work in point-to-point manner are ineffective as well since they are very prone to get stuck in some sub-optimal solution when searching huge space with many local optima.

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 550–559, 2013.
© Springer-Verlag Berlin Heidelberg 2013

On the contrary, evolutionary algorithms (EAs) are population-based search and optimization techniques that have been used for solving hard optimization problems of the black-box type. Unlike the single-state heuristics, the search direction is adjusted using the information accumulated over all candidate solutions of the population in each generation step of the EA run. Thus, the EAs are more resistant to getting trapped in a local optimum. They are also resistant to noise in the evaluation function, which is very important for this particular optimization domain.

There have been many studies devoted to the evolutionary design of systems for automated gait generation and gait optimization of biped, quadruped, and hexapod robots [3]. Several types of gait representation has been used by evolutionary-based gait generators. Genetic Programming and Grammatical Evolution evolve directly the functions defining the joint angle trajectories [6]. One example of this approach is in the work of Ivan Tanev who used it to create controllers for artificial snakes [7]. Another approach to this problem is evolution of Central Pattern Generators (CPG) [2]. Those are type of neural network with the ability to generate rhythmic patterns.

The research presented in this paper is based on modular robotic creatures that are composed of a number of simple cubic-shaped robotic blocks. The robotic blocks are endowed with a movable arm and several slots that they use to connect to each other (realizing joint-like connections) to form complex structures. Joint angles are controlled by functions of a single input, time, that return desired joint positions at discrete time steps. Thus, the generation of the gait consists in finding a set of functions (a single function for each joint) that make the whole robotic creature to move in a desired way. We assume multi-legged robots that exhibit features of symmetry and module repetition. Particularly, a quadruped robot was investigated in this work.

In this work we consider just simulations of the robot and its gaits that are carried out using a simulation platform Sim, based on ODE physics simulator[1], which has been developed within the SYMBRION and REPLICATOR[2] projects [4], focused on an application of evolutionary and swarm techniques in robotics.

The primary goal of our research was to design an effective and efficient evolutionary-based system for automated generating of robot gaits and experimentally evaluate its performance. To attain the goal, a co-evolutionary system was proposed that resolves the whole task by decomposing it into:

- The evolution of a single-leg motion pattern, i.e. evolution of functions controlling movement of all joints of a single leg. This part is realized by genetic programming.
- The evolution of the coordination strategy that optimally deploys the evolved single-leg motion pattern to all legs of the robotic creature. This part is realized by genetic algorithm as the coordination strategy is represented by a linear vector of control parameters that are to be tuned.

[1] `www.ode.org`
[2] `http://www.symbrion.eu/tiki-index.php`

The co-evolutionary aspect of this approach arises from the fact that the single-leg motion patterns are coupled with coordination strategies for the purpose of assessing their quality.

Series of proof-of-concept experiments were carried out in order to analyse capabilities of the proposed approach. Its performance was compared to the performance of a simple GP approach that simultaneously evolves functions to all of the joints of the whole creature. Results show that the co-evolutionary approach delivers significantly better results than the compared GP approach.

This paper is structured as follows. In section 2, characteristics of robots simulated in the simulation platform Sim are described. Section 3 describes the proposed co-evolutionary approach for automatic robot gait generation. Section 4 describes the experimental proof-of-concept scenario and presents the achieved results. Final section concludes the paper and lists further extensions of the presented approach.

2 Simulation Platform and Experimental Robot

In this work, we use a simulation platform Sim [8], not the real physical robots. However, the simulation platform, which is based ODE physics simulator, provides very realistic and accurate experimental environment.

In Sim, robots composed of a number of simple cubic-shaped robotic blocks, see Fig. 1a), can be investigated. Each block consists of the main body and one movable arm. This arm has its axle in the center of the body and can be moved between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. While the main body of the block has wheels which can be used to move it on the flat ground, those are not used as a means of motion. Instead, the blocks are endowed with slots (three of them on the main body and one is on the movable arm) that enable them to connect to each other and form more complex robots capable of walking motion.

By connecting a slot on the movable arm of one block to the slot on the main body of another block a joint-like functionality is realized. Motors moving with arms are controlled by input signals which express a desired angle of the arm. This means that the motor tries to move the arm to the desired angle, however it may fail to reach exactly the desired position due to the physical constraints.

In this work we focus on multi-legged robots that exhibit features of symmetry and module repetition. Particularly, a quadruped robot was investigated, see Fig. 1b). This structure of the robot was fixed for all experiments as the search for ideal morphology is not part of this work. This robot is assembled from 17 blocks, five of these blocks form a torso of the robot and remaining twelve blocks are used for four identical legs. Note that the movable arms do not realize spherical joints, they can move only in one plane, either the vertical or the horizontal one. Thus the connection types chosen for individual joints of the robot determine the character of gait the robot can perform. The robot used in our experiments is designed to walk sideways just as crabs do.

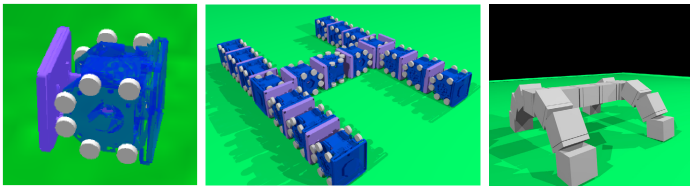


Fig. 1. A single robotic block used as a building block of robotic creatures, the complete robot in starting position (i.e. lying on the floor) and a higher-level abstraction of the same robot in standing position.

3 Proposed Approach to Automatic Motion Pattern Design

In this work we assume multi-legged robots that exhibit symmetry and module repetition properties and propose a gait generation algorithm that makes use of these properties to decompose the whole task in order to efficiently evolve gaits of such complex robots. There are two modules, genetic programming module (*GP-module*) and genetic algorithm module (*GA-module*), involved in the algorithm, each devoted to evolution of a specific trait of the resulting gait:

- The *GP-module* is responsible for evolution of a single-leg motion pattern, i.e. evolution of functions controlling movement of individual joints of a single leg and their coordination. A standard tree-based GP is used to realize this module.
- The *GA-module* is responsible for evolution of a coordination strategy that optimally deploys the single-leg motion pattern, evolved by the *GP-module*, to all legs of the robot. This module is realized by genetic algorithm as the coordination strategy is represented by a linear vector of control parameters that are to be tuned.

The two modules co-evolve simultaneously in a sense that the single-leg motion patterns are coupled with coordination strategies for the purpose of assessing their quality. So, when evaluating one particular individual of *GP-module* some individual of the *GA-module* has to be assigned in order to make a complete gait, and vice versa. The motivation for this co-evolutionary approach is to let the system seek for a good single-leg motion pattern that could be re-used, with certain adjustment for left/right and fore/hind leg, multiple times for all legs of the complete robot [1]. Clearly, this approach could be used only if all legs of the robot have the same structure and the motion patterns they perform are expected to be similar[3].

[3] Note, the joints connecting the blocks of the robot's torso are fixed, they are not included into the evolved gait control strategy.

3.1 Genetic Programming Module

Representation. Given that each leg is composed of k connected blocks, where each of the k joints is controlled by its own function, the single-leg motion pattern is then represented by k trees which use the following sets of terminals and non-terminals:

- non-terminals: +, -, unary -, sin, cos, *, /, max, min
- terminals: input variable time t, π, real-valued constants $r = \langle -1, 1 \rangle$

Genetic Operators. Standard GP subtree-swapping crossover operator was implemented. On the other hand, no mutation operator was used. To select individuals for reproduction, the tournament selection method was chosen.

To reduce the bloat and increase the computational efficiency of the algorithm, a bloat control method called *Proportional Tournament* [5] was used. In this method, a proportion of tournaments, given by parameter R, are based on parsimony and the remaining tournaments, $1 - R$, are based on fitness. This means that either well-fit or small tree wins the tournament.

Evolution Model and Initialization. This algorithm uses generational evolution model. The first generation is initialized by *ramped-half-and-half* method with the maximal depth limit of 5.

3.2 Genetic Algorithm Module

Representation. This module evolves coordination strategies that determine the way a particular single-leg motion pattern is deployed to all legs of the robot. The coordination strategy is given by the following parameters defined for each leg:

- *Direction*, d, specifies whether the single-leg motion will be applied in the direct or its reverse mode. If the reverse mode is chosen for a particular leg then the input time parameter passed into its joints functions is negated so that the leg performs a motion which is a "mirrored" version of the original single-leg motion. This is important for the crab like motion as it allows legs on one side of the robot to pull and legs one the other side to push. The direction parameter is represented by a single bit for each leg.
- *Phase*, φ, specifies phase shift in legs movement. This makes it possible to have different legs in different phases of the gait at the same time. This parameter is represented by a real number from interval $\langle 0.0, 1.0 \rangle$, where the interval covers all possible shifts within one period of the corresponding single-leg motion.

The final structure of the chromosome representing a coordination strategy for n legs is as follows

$$[d_1 \dots d_n \mid \varphi_1 \dots \varphi_n]$$

where the direction bits are grouped in the head part and the phase real numbers in the tail part of the chromosome.

Genetic Operators. The representation described above requires a crossover operator that operates differently for the two parts of the chromosome.

Standard one-point crossover is used for the binary string of direction bits. For the real-valued tail part of the chromosome, a variant of the *Blend crossover* was implemented. Given two parents $p1$ and $p2$, the offspring phase value at position i, $o_{\varphi i}$, is taken at random (with uniform distribution) from the following set of possible choices

$$o_{\varphi i} \in \{p1_{\varphi i}, p2_{\varphi i}, (p1_{\varphi i} + p2_{\varphi i})/2, min(p1_{\varphi i}, p2_{\varphi i}) - 0.5 * range, max(p1_{\varphi i}, p2_{\varphi i}) + 0.5 * range\}$$

where $range = abs(p1_{\varphi i} - p2_{\varphi i})$. Every newly generated value $o_{\varphi i}$ is further adjusted to fall into the interval of admissible values $\langle 0.0, 1.0 \rangle$.

For the binary direction parameters, simple bit-flip mutation operator was used. When a phase parameter is selected for mutation, it is replaced by a new value randomly generated from $\langle 0.0, 1.0 \rangle$. Tournament selection is used to select parents for crossover and mutation.

Evolution Model and Initialization. This algorithm uses a steady-state evolution model where the newly generated individual replaces the worst individual in the population. The first generation is initialized randomly.

3.3 Co-evolution of *GP-module* and *GA-module*

As mentioned at the beginning of this section, the co-evolution of *GP-module* and *GA-module* is based on pairwise relationships between individuals from these two populations. These relationships are necessary for assessing the quality of the individuals so that when evaluating an individual from *GP-module* its counterpart from *GA-module* is used to make up the whole gait strategy, and vice versa. Below, the fitness function used to assess evolved gait strategies and the scheme used for maintaining the pairwise relationships are described.

Fitness Function. The same minimization fitness function is used to assess individuals from *GP-module* and *GA-module*. It takes into account two aspects of the evaluated gait - (i) a *distance covered* by the robot in desired direction and (ii) a *body posture* of the robot during its movement.

The covered distance term is measured as the distance of the central block of the robot's torso from desired destination point. The body posture term is defined as a penalty punishing robots that show tendency to drag its body on the ground. This is realized so that a minimum height of the central block of the robot is specified and then certain value is added to the final penalty for each step of the whole simulation of the robot's movement[4] the robot made in lower posture. Note, the penalty is in fact very high, hence strongly encouraging evolution of walking gaits instead of crawling ones.

[4] The simulations are carried out in discrete steps, with frequency 125 steps per second.

Pairing off Individuals of *GP-module* and *GA-module*. At the beginning of the run, individuals of the *GP-module* are randomly paired off with individuals of the *GA-module* so that each *GP-module* individual has exactly one counterpart in *GA-module*. On the other hand, the *GA-module* individual can have arbitrary number of links (0 to many) to individuals in *GP-module*.

Then, the co-evolution proceeds by phases, altering M generations of *GP-module* and N generations of *GA-module* in each phase. When the *GP-module* generates a new population, each newly generated individual inherits the relationship with the *GA-module* individual from one of its parents. This way it is ensured that every individual in *GP-module* has always assigned its counterpart in the *GA-module* for evaluation purposes.

When *GA-module* evolves its population, each newly generated individual is evaluated with several candidates from *GP-module* and the best value is set as its fitness. Moreover, it may happen that some of its individuals has lost its counterpart in the *GP-module* (because they were replaced by new individuals in some of the previous generations of *GP-module*). In this case, the fitness of such unpaired *GA-module* individual is penalized. This way individuals with no link to *GP-module* can still compete between each other but will always loose against those that are linked to someone in *GP-module*. Also, whenever an individual from *GA-module* is deleted it is important to find a new *GA-module* individual for all *GP-module* individuals which were connected to it.

4 Experimental Evaluation

Series of proof-of-concept experiments were carried out in order to analyze capabilities of the proposed approach. Its performance was compared to the performance of a simple GP approach that does not make use of the problem decomposition. Instead, it uses just the *GP-module* to simultaneously evolve functions to all of the joints of the whole robot. In particular, the simple GP approach evolves individuals, each representing twelve functions (4 legs times 3 joints).

4.1 Experimental Scenario

To evaluate performance of implemented method a simple experiment was set up. Gaits for the robot described in Section 2 were evolved, where the environment was just a flat surface without any obstacles. When simulating the robot with a candidate gait, the robot was always placed in the center of the simulation arena in laying position. First, the robot was given a fixed amount of simulation steps to start moving, hopefully to stand up, and then it was let to walk as far as possible for specified amount of time (10 seconds in this case). Finally, the distance of its final position from the specified target point was measured.

4.2 Experimental Set-up

Control parameters of the co-evolutionary algorithm were set as follows:

- *GP-module*: population size 500, generations 200, tournament size 7;
- *GA-module*: population size 50, tournament size 3;
- synchronization: start-up phase of 20 generations of the *GP-module* and then repeatedly carrying out 2 generations of *GP-module* followed by 30 newly generated individuals in the *GA-module*;

Control parameters of the *GP-module* of the simple GP approach were set as follows: population size 600, number of generations 250. This configuration ensures that the simple GP algorithm should compute at least as many fitness evaluations (i.e. simulations) as the co-evolutionary one.

4.3 Results

All control functions evolved with both algorithms were able to make the robotic creature walk or at least move in some way. The resulting paths are shown in Figure 2 with the compared simple GP algorithm on the left side and the proposed co-evolutionary approach on the right side. While both approaches led to some good and some poor solutions, the co-evolutionary approach achieved significantly better results than the compared approach.

The mean best distance from the final robot's position to the target point calculated from solutions of ten independent runs was 35.98 in case of the co-evolutionary algorithm and 38.67 for the compared one, respectively (note, the less the better). The Mann-Whitney rank sum test gives significant evidence ($p < 0.05$, two-tailed) for the observed difference. The best solution reached by the co-evolutionary algorithm has value of 33.08 and the best solution reached by the compared algorithm has value of 35.16.

Second important observation is that the co-evolutionary approach produces neat gaits in contrary to the ones produced by the simple GP approach. Simply

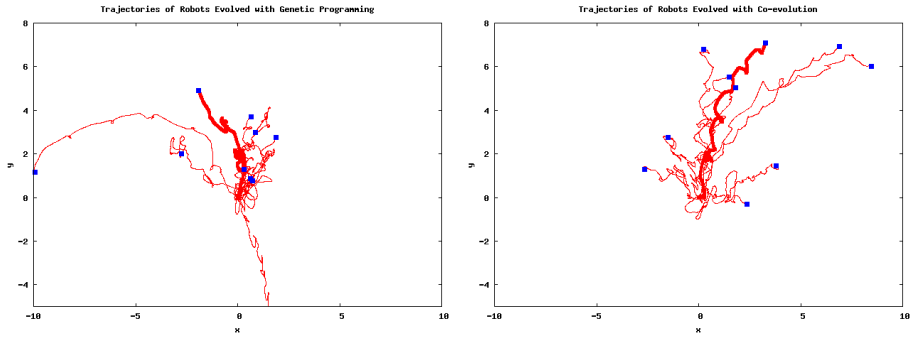


Fig. 2. Trajectories of robots with control functions evolved by both algorithms. Simple GP approach on the left, co-evolutionary approach on the right.

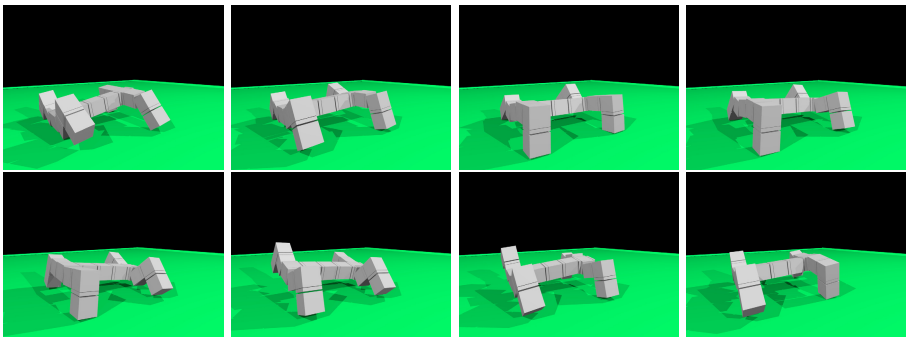


Fig. 3. Example of the gait produced by the co-evolutionary approach

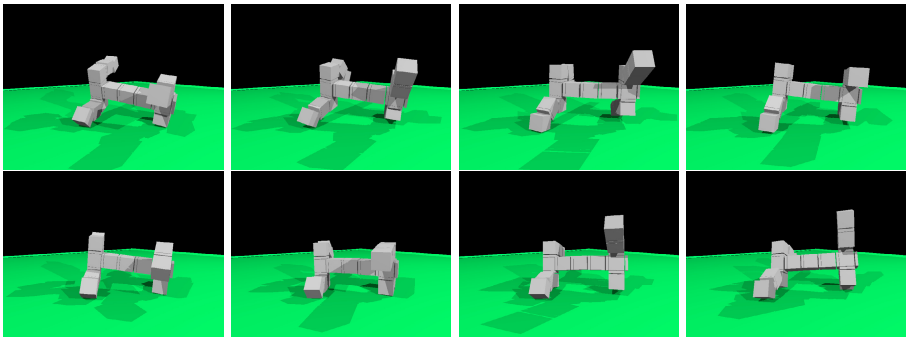


Fig. 4. Example of the gait produced by the simple GP approach

said, the co-evolutionary robots walk while the GP robots just get somewhere somehow in very uncoordinated way, see Figures 3 and 4.

For example it was not uncommon for the simple GP to develop gaits which only used 3 legs and held the remaining one in the air or dragged it on the ground. Such gaits even though they are leading to some motion cannot be considered fast nor efficient gaits.

One of the GP evolved gaits even made the robot walk backwards as can be seen in Figure 2. This might happen due to the fitness function which strongly forces the robot to walk with elevated body and not to crawl on the ground (note that all crawling movements are rewarded less than any movement with the elevated body). It is evident, that in this case the GP approach was only able to evolve "well-rewarded" walking gait that made the robot walk backwards.

5 Discussion and Future Work

In this paper we present a co-evolutionary approach for automatic generation of robotic gait. It simultaneously evolves single-leg motions and synchronization schemes that optimally deploy the single-leg motion to all legs of the robot.

Our results indicate that this method of decomposition of the problem can generate better motion patterns than simple GP approach. The gaits found by the co-evolutionary approach outperform the ones found with the simple GP in terms of the distance walked towards the desired target point as well as in terms of the orderliness and efficiency of the observed motion patterns.

There are a number of possible further extensions. One of them is evolution of multiple motion primitives like left and right turn and forward and backward walking. For example, it is assumed that when given an efficient forward walking gait the co-evolutionary approach should be able to change the direction of walking just by evolving new leg-synchronizing strategy in *GA-module* for the fixed single-leg motion patterns. Those walking primitives can be further used to navigate a robot along some desired path.

Another area of interest is evolution of gaits for more challenging environments such as sloppy surfaces or walking up and down stairs. Since the most time- and resource-consuming computations are related to the simulations it might be useful to design the evaluation procedure in a staged manner so that the candidate gait first undergoes simple and short simulations and only if it passes them it proceeds to the main simulation that returns the objective value.

Acknowledgment. Jiří Kubalík was supported by the research program No. MSM 6840770038 "Decision Making and Control for Manufacturing III" of the CTU in Prague and Jan Černý was supported by the Grant Agency of the CTU in Prague, grant No. SGS12/145/OHK3/2T/13.

References

1. Cerny, J.: Evolutionary design of robot motion patterns (June 2012), http://cyber.felk.cvut.cz/research/theses/papers/226.pdf
2. Crespi, A., Ijspeert, A.: Online optimization of swimming and crawling in an amphibious snake robot. IEEE Transactions on Robotics 24(1), 75–87 (2008)
3. Gong, D., Yan, J., Zuo, G.: A review of gait optimization based on evolutionary computation. In: Applied Computational Intelligence and Soft Computing 2010 (2010)
4. Kernbach, S., Meister, E., Schlachter, F., Jebens, K., Szymanski, M., Liedke, J., Laneri, D., Winkler, L., Schmickl, T., Thenius, R., et al.: Symbiotic robot organisms: Replicator and symbrion projects. In: Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems, pp. 62–69. ACM (2008)
5. Panait, L., Luke, S.: Alternative Bloat Control Methods. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 630–641. Springer, Heidelberg (2004)
6. Seo, K., Hyun, S.: Genetic programming based automatic gait generation for quadruped robots. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 293–294. ACM (2008)
7. Tanev, I., Shimohara, K.: Co-evolution of active sensing and locomotion gaits of simulated snake-like robot. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO 2008, pp. 257–264. ACM, New York (2008), http://doi.acm.org/10.1145/1389095.1389135
8. Vonasek, V., Fiser, D.: Sim: a general purpose 3d robotic simulator (2012), http://sim.danfis.cz/

A Comparison between Different Encoding Strategies for Snake-Like Robot Controllers

Dámaso Pérez-Moneo Suárez and Claudio Rossi

Universidad Politécnica de Madrid,
Centre for Automation and Robotics, Madrid, Spain
damaso.psuarez@alumnos.upm.es, claudio.rossi@upm.es

Abstract. In this paper, we present the results of the tests we have performed with different encoding strategies for evolving controllers for a snake-like robot. This study is aimed at finding the best encoding for on-line learning of basic skills, such as locomotion (both free and directed to an objective) and obstacle avoidance. The snake moves in a virtual world, which realistically simulates all the physical conditions of the real world. This is the first step of our research on on-line, embedded and open-ended evolution of robot controllers, where robots have to learn how to survive during their lifetime, and occasionally mate with other robots. A simple (1+1) evolutionary strategy has been adopted for lifetime learning. The results of the tests have shown that the best results, tested on the locomotion skills, is the 'He1Sig' controller, that uses a different set of parameters for each segment of the snake but only one mutation rate, common to all parameters, that is encoded in the chromosome and therefore undergoes evolution itself.

Keywords: Evolutionary robotics, embodied evolution.

1 Introduction

The aim of the research presented here is to develop new evolutionary strategies for generating robot controllers for real, physical robots. The scenario we are working on is an ecosystem of robots that evolve both their physical structure (body) and control software (mind). In this ecosystem, like in the real world, individuals have to learn some basic skill to survive, or improve the abilities that they have inherited by their parents. They will have to learn to find energy sources, without which they would eventually starve to death. Occasionally, they will meet and mate. New offspring will be generated by crossing over their genetic material. These will be at all effects *new* individuals: we assume that an instrument for their physical implementation exist. This shall be an automated production mechanism that, following their genetic blueprint, would build the newborn individuals, by assembling basic parts ("organs") or even by producing them on-the-fly by 3D printing technology [6].

In this work, we focus on *lifetime learning*, as a necessary step towards the kind of system described above. Our purpose is to implement lifetime learning by

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 560–568, 2013.
© Springer-Verlag Berlin Heidelberg 2013

an on-line and embedded (on-board) evolution process of the robots' controllers. This has the aim of improving their survival (and hence reproduction) possibilities. Lifetime learning is thus mainly aimed at improving mobility, foraging and mate finding skills. Clearly, mobility is a key feature for the achieving food and mates. Therefore, in this work we focus on this fundamental aspect.

The robot ecosystem we work with exists in a virtual world, simulated adopting realistic 3D simulators, that faithfully reproduce both the real world physics and the robots' *kinematics* and *dynamics*, as well as the physical interactions with the world's objects.

By doing this we obviously aim at reducing the *reality gap*. But more importantly, there is a growing feeling in the research community that "matter matters"[11]. That is, the body, its shape, and its physical interaction with the environment have an impact on the way the mind is shaped. Therefore, we include realistic physics in our simulated ecosystem.

Given the realistic physical simulations, the computational cost of the experiments is very high[1]. For this reason, we have adopted a simple structure such as a chain of modules, obtaining a snake-like robot, which is the simplest structure conceivable. Each segment of the snake's body is connected to the others with an actuated rotational joint that allows undulatory movements of the snake in the horizontal plane. Each actuator moves with a sinusoidal function (see below). Thus, each actuator is controlled by three parameters: rotation speed, rotation amplitude and rotation phase with respect to the head. By including an additional degree of freedom at the joints, a wide variety of movements in 3 dimensions could be generated, also for more complex morphologies (e.g. in form of H, that would resemble a four-legged robot).

Related Work

There is a considerable body of work on controllers for snake-like robots controllers. Undulatory motion can be generated e.g. by central pattern generators [9], Neural Networks [7,4,12,2], or modulated architectures [1,14,10].

Central pattern generators and neural networks are both powerful techniques, but have the drawback of needing too much information (think, e.g., at the number of connection weights that the have to be evolved).

It must be pointed out that the most usual configuration for evolving robot controllers is off-line and off-board: an evolutionary algorithm is run in an external computer, and controllers are then deployed into the robots (either real or simulated) to be evaluated. This is the case e.g. of the one of the most known evolutionary robotics works, the evolving virtual creatures of C. Sims [12] and J. Bongard [2].

Recently, on-line learning has also been proposed in the literature (see e.g. [13], where virtual agents are evolved in continuous time, but without on-line learning). In particular, Eiben et al. [5] used a embodied on-line and on-board distributed evolution. Each robot have a controller population that evolve inside

[1] With 6 robots in the arena, simulation time was $t = 0.06\times$ w.r.t. real time.

it, and at given times robots exchange controllers with others. This strategy is called *island method*. Here, we adopt a similar strategy, in an minimal configuration, where each island has just one robot controller, an approach similar to the one proposed in [3], where a (1+1)-online Evolutionary Algorithm is proposed.

The main differences with this work lays in the genotypes (robot controllers) representation (Neural Networks vs. periodic functions) and in that in this work we focus at comparing different encodings for the proposed genotype.

Snake Motion

We model the snakes movement using the known "serpenoid" functions described by Hirose (see e.g. [8]):

$$\phi_i(t) = 2\alpha \sin(\omega t + (i-1)\beta) \tag{1}$$

The term $\phi_i(t)$ is a sinusoidal function that determines the position of joint i between two body segments, and n is the number of body segments of the snake. The parameters ω, α and β determine the shape of the serpentine curve realized by the snake-like robot, defining respectively angular speed, amplitude and phase of the oscillation of the joints.

Using this 'a priori' knowledge of the snake's motion, we can achieve a quicker evolution, since the dimension of the search space is considerably reduced w.r.t. other approaches.

The question we address here is which is the best encoding for the serpenoid curve parameters, as far as locomotion efficiency obtained is concerned. This is a key factor that affects learning time and thus the survival possibilities of the robots. Choosing a good chromosome encoding is a key question, since it affects both the quality of the solutions found and the evolution time.

The rest of this paper is structured as follows. Section 2 describes in details our evolution strategy, and Section 3 illustrates the results of the simulations carried out. Finally, Section 4 discusses the results obtained and future work.

2 Controllers Encoding

Equation 1 can be represented in two ways. In the first way, each joint has independent parameters (Eq. 2). In the second way, joints parameters are expressed as a (constant) difference w.r.t. the previous joint (Eq. 3).

$$\phi_i(t) = A_i \sin(\omega_i \cdot t - \varphi_i), \quad i = 1 \,..\, n-1 \tag{2}$$

$$\phi_i(t) = (A_0 - i \cdot \Delta A) \cdot \sin((\omega_0 + i \cdot \Delta\omega) \cdot t - i \cdot \Delta\varphi), \quad i = 1 \,..\, n-1 \tag{3}$$

We call equation (2) *'heterogeneous'*, because in this case each module has its own value for the amplitude, phase and frequency. This will need a longer chromosome, of $3 \cdot (n-1)$ genes, n being the number of body segments. Equation (3) implies that some regularities must exist between the movements of the joints,

and is called *homogeneous* because it is assumed that there is a regular difference between the amplitude, phase and frequency between the modules. Therefore it will need just three values, the difference in amplitude, phase and frequency between each module $\Delta A, \Delta\omega, \Delta\varphi$. In Eq. (3) the base values corresponding to the first module adopted are $A_0 = 1$ and $\omega_0 = 0$.

The two equations proposed above represent the two extremes of a wider range of choices, where groups of modules can share the same parameters.

In either case, the chromosome is complemented with mutation rate information. We considered different ways to express the mutation rate:

- constant mutation rate for all genes [WSig]
- one mutation rate for all genes, encoded into the chromosome [1Sig]
- one mutation rate for each gene, encoded into the chromosome [NSig]

Hence, we tested six different configurations, summarized in Table 1.

Table 1. Parameters used in the experiments. Mutation rates μ and μ_i are encoded in the chromosome, $(i = 1 \ldots n-1, n = 5)$.

Experiments parameters						
Parameter	He1Sig	HeNSig	HeWSig	Ho1Sig	HoNSig	HoWSig
Representation	Heterogeneous			Homogeneous		
Population size	1	1	1	1	1	1
Generations	500	500	500	500	500	500
Mutation rate	μ	μ_i	0.1	μ	μ_i	0.1

3 Experiments

The virtual world has been created using the well known 3D simulation software 'Webots'. A snakelike modular robot (see Figure 1) composed of five segments modules has been used to test the different configurations described above.

The simulation arena was divided into six smaller areas, were a snake for each configuration has been tested (Fig. 2). Snakes can move freely in the arena, and each controller is tested for 10 seconds.

To avoid biasing form the initial position, each 10 seconds, when the snakes change the controller, they are relocated in the initial position.

Before running the simulations, an 'ideal' snake adopting Eq. 1 was tested in order to measure the distance that it could travel. By ideal, we mean with optimal values for the functions' parameters. This can be done precisely because we have an indication of the optimal motion for the given structure [8], an information that it would not be easy to obtain for other structures.

This data, corresponding to the term 11.62, was introduced in the fitness defined by Equation 4 to normalize its values.

In order to test locomotion, the fitness function that was used is:

$$Fitness = \frac{Dist_{accum}}{11.62} * \frac{dist_{straight}}{dist_{accum}} = \frac{dist_{straight}}{11.62} \tag{4}$$

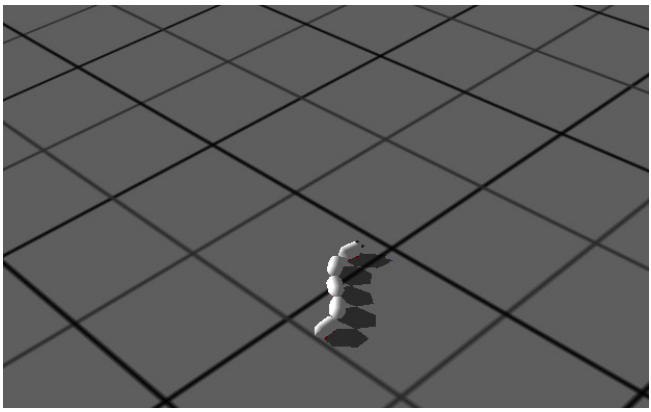


Fig. 1. The snake-like robot architecture employed it is composed of 5 identical body segments

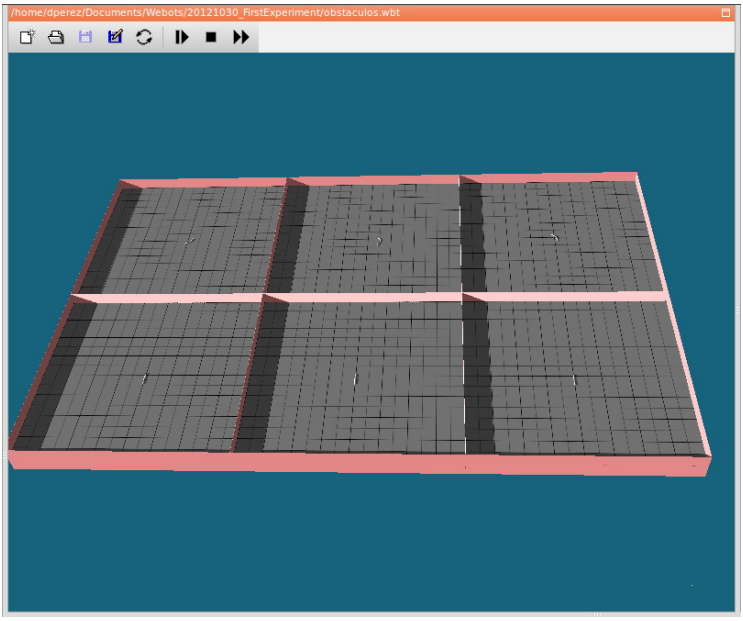


Fig. 2. Arena with 6 Snakes in Webots

Table 2. Experiments results of each chromosome

	Controller					
	He1Sig	HeNSig	HeWSig	Ho1Sig	HoNSig	HoWSig
Best	0.899	0.849	0.847	0.626	0.533	0.513
Average	0.767	0.675	0.690	0.225	0.275	0.282
Std. dev.	0.163	0.163	0.178	0.116	0.068	0.090

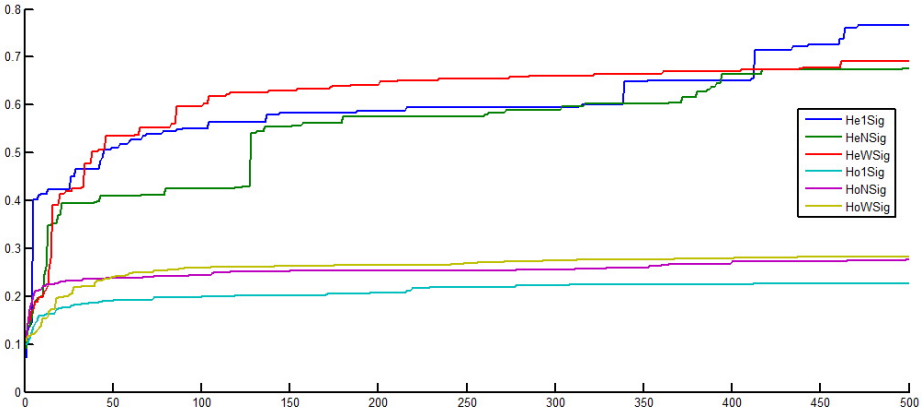


Fig. 3. Average Best Fitness of the different encodings

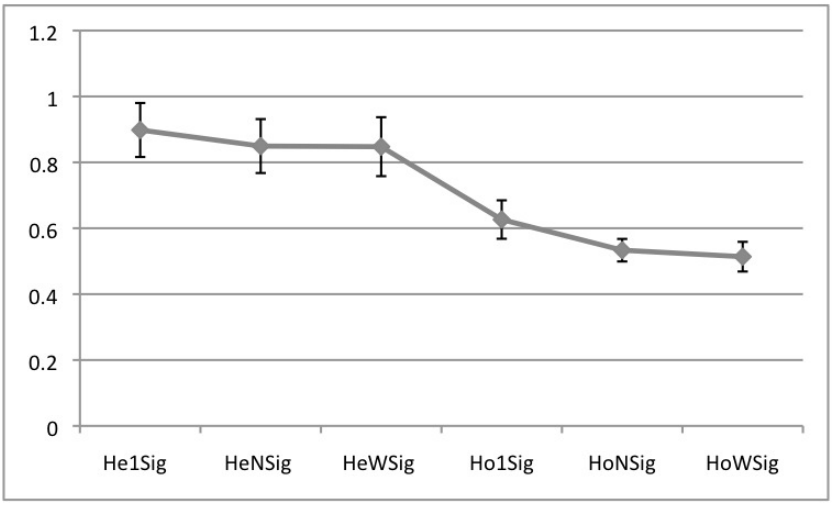


Fig. 4. Best fitness obtained for the different configurations

where $dist_{accum}$ is the straight motion in relation to the initial position of the snake, and $dist_{straight}$ is the motion in relation to the orientation of the snake (longitudinal). The ratio $dist_{straight}/dist_{accum}$ expresses how much of the distance travelled was actually due to serpentine locomotion, since the snake could roll over itself and thus travel considerable distances. Thus, fitness is normalized in such a way that a value of 1 means ideal motion[2].

[2] Clearly, in Eq. 4 it is assumed that the serpentine motion is the best one. Other kind of motions could in principle have a fitness greater that 1, although this case has never happened in the simulations.

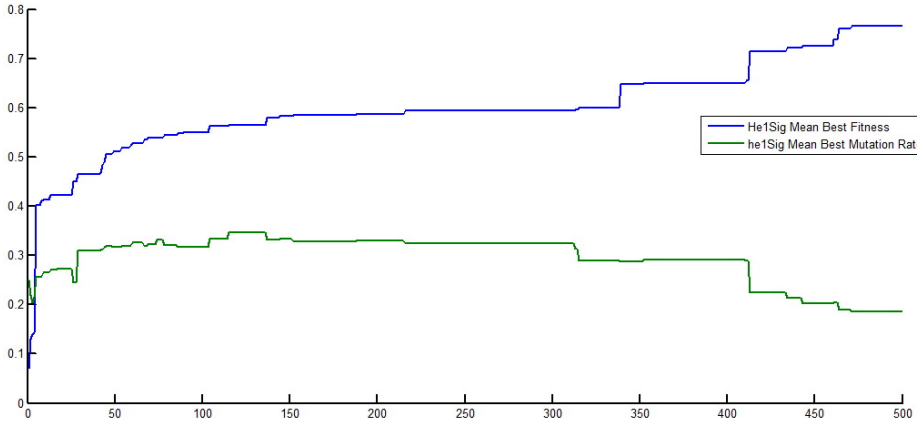


Fig. 5. Mean Fitness and Mean Mutation of 'He1Sig' chromosome

For each different encoding, 20 different runs were performed. Each snake was allowed to evolve for 500 generations.

Figure 3 shows a comparison of the evolution of the best fitness of the different encodings, while Figure 4 and Table 2 summarise the numerical results for all the experiments. As it can be noticed, the overall best results have been obtained with the the 'He1Sig' encoding (a different set of parameters for each joint, i.e. $3(n-1)$ genes, with a common mutation rate encoded into the chromosome for all the genes). However, all three heterogeneous controllers perform significantly better than the homogeneous encodings.

It is also interesting to see the relationship between the fitness and the mutation rate in the controller *'He1Sig'*. In Figure 5, the average fitness and mean mutation rate of the *'He1Sig'* controller are shown. It is easy to see how the mutation rate is bigger at the beginning of the run. As the controller evolves and gets better fitness values, the mutation rate gets smaller.

4 Discussion and Conclusions

The performance of the best configuration has been satisfactory, as it could reach values very close to the ideal value of 1.

However, the numerical results obtained from the experiments, are somehow unexpected. In fact, the Hirose serpenoid curve (Eq. 1) defines the ideal movement, where the oscillations of the joints are synchronised in such a way to obtain a smooth undulatory motion of the whole body, resulting in the best advancing motion. This has only three parameters that are the same for all links. Here, it appears that the 'Heterogeneous' representations obtain the best results. These, having an individual set of parameters for each joint, were expected to perform worst, since the search space is much bigger, and the oscillation of each joint should orchestrate with all the other ones in order to obtain a smooth oscillation of the whole snake body. In other words, the task to learn is more difficult.

Note also that the encodings that which has mutation rate encoded in the chromosome perform better, with the added advantage of not needing parameter tuning.

In conclusion, we have verified that in a relatively short lifespan the individuals encoded with the heterogeneous system are capable of learning the locomotion task. This is a very good result since, as stated earlier, locomotion is the basis for foraging and mate finding, which will be the next tasks that we plan to include in the lifetime learning tests of the robot snakes.

Note also that the heterogeneous encoding is more suited for more complex robot morphologies, and hence it can be claimed that the results obtained will be generalizable to other shapes.

As a next steps of our work, when the robots will be able to learn to effectively learn all the basic skills, the mating will be implemented, obtaining the final ecosystem, where individuals will be generated by asynchronous mating. Time will not be discrete, in form of "generations", as it is common in evolutionary algorithms, but continuous real time. New individuals will enter the world and will survive and reproduce according to their skills, part of which will be inherited, and part developed trough learning. The life span of the individuals will only be determined by their ability to find food, although both ageing mechanisms and grace period for newborn individuals shall be devised. Population size will thus be variable, only limited by the amount of resources available in the world. but continuous real time. The major future challenge of this work will be extending the proposed life cycle including evolvable morphologies for the individuals.

References

1. Becerra, J.A., Bellas, F., Duro, R.J., Lope, J.D.: Snake-like behaviors using macroevolutionary algorithms and modulation based architectures
2. Bongard, J.: Morphological change in machines accelerates the evolution of robust behavior. Proceedings of the National Academy of Sciences 108(4), 1234–1239 (2011)
3. Bredeche, N., Haasdijk, E., Eiben, A.: On-Line, On-Board Evolution of Robot Controllers. In: Collet, P., Monmarché, N., Legrand, P., Schoenauer, M., Lutton, E. (eds.) EA 2009. LNCS, vol. 5975, pp. 110–121. Springer, Heidelberg (2010)
4. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperNEAT generative encoding. In: Congress on Evolutionary Comutation (CEC) (May 2009)
5. Eiben, A., Haasdijk, E., Bredeche, N.: Embodied, On-line, On-board Evolution for Autonomous Robotics. In: Levi, P., Kernbach, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution.. Cognitive Systems Monographs, vol. 7, pp. 361–382. Springer (2010)
6. Eiben, A., Kernbach, S., Haasdijk, E.: Embodied artificial evolution. Evolutionary Intelligence 5, 261–272 (2012)
7. Haasdijk, E., Rusu, A.A., Eiben, A.E.: HyperNEAT for Locomotion Control in Modular Robots. In: Tempesti, G., Tyrrell, A.M., Miller, J.F. (eds.) ICES 2010. LNCS, vol. 6274, pp. 169–180. Springer, Heidelberg (2010)

8. Hirose, S., Morishima, A.: Design and control of a mobile robot with an articulated body. Int. J. Robot. Res. 9, 99–114 (1990)
9. Ijspeert, A.: Central pattern generators for locomotion control in animals and robots:a review. Preprint of Neural Networks 21/4, 642–653 (2008)
10. Lal, S., Yamada, K., Endo, S.: Evolving motion control for a modular robot. In: Ellis, R., Allen, T., Petridis, M. (eds.) Applications and Innovations in Intelligent Systems XV, pp. 245–258. Springer, London (2008)
11. Pfeifer, R., Bongard, J.C.: How the Body Shapes the Way We Think. A New View of Intelligence. MIT Press (2007)
12. Sims, K.: Evolving virtual creatures. In: Annual Conference Series, pp. 15–22 (July 1994)
13. Stanley, K., Bryant, B., Miikkulainen, R.: Real-time neuroevolution in the nero video game. IEEE Transactions on Evolutionary Computation 9(6), 653–668 (2005)
14. Weel, B., Haasdijk, E., Eiben, A.E.: The Emergence of Multi-cellular Robot Organisms through On-Line On-Board Evolution. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 124–134. Springer, Heidelberg (2012)

MONEE: Using Parental Investment to Combine Open-Ended and Task-Driven Evolution

Nikita Noskov, Evert Haasdijk, Berend Weel, and A.E. Eiben

Vrije Universiteit Amsterdam, The Netherlands
n.k.noskov@gmail.com, {e.haasdijk,b.weel,a.e.eiben}@vu.nl

Abstract. This paper is inspired by a vision of self-sufficient robot collectives that adapt autonomously to deal with their environment and to perform user-defined tasks at the same time. We introduce the MONEE algorithm as a method of combining open-ended (to deal with the environment) and task-driven (to satisfy user demands) adaptation of robot controllers through evolution. A number of experiments with simulated e-pucks serve as proof of concept and show that with MONEE, the robots adapt to cope with the environment and to perform multiple tasks. Our experiments indicate that MONEE distributes the tasks evenly over the robot collective without undue emphasis on easy tasks.

1 Introduction

The work presented in this paper is inspired by a vision of autonomous, self-sufficient robot collectives that can cope with situations unforeseen by their designers. An essential capability of such robots is the ability to adapt their controllers in the face of challenges they encounter in a hands-free manner, "the ability to learn control without human supervision," as [14] put it.

One approach to solve this issue uses evolution as a force for *adaptation*, rather than as "just" an algorithm for *optimisation*. This dichotomy has been noticed early in the history of evolutionary computing, [4]. Since then, these two attitudes have became dominant in different areas. Optimisation is the primary goal in Evolutionary Computing [6], while evolution as a driver of adaptation is typical in Artificial Life (ALife) [23]. In a common ALife setting, agents, possibly (simulated) robots, populate a world and the one that can cope with its environmental challenges will survive and reproduce. In systems like this, there need not be any objective function to be optimised, nor a centrally orchestrated evolutionary selection–reproduction loop. Instead, evolution is driven by a decentralised, asynchronous process of mate selection and reproduction and purely environmental selection that gives a reproductive advantage to well-adapted individuals. Such open-ended approaches are slowly finding their way into evolutionary robotics, e.g. the mEDEA algorithm[2] and Bianco and Nolfi's work [1].

Of course, an adapting robot collective must also serve the purpose of its designers: it must satisfy human preferences and tackle particular tasks. Evolutionary robotics has traditionally focussed exclusively on this latter aspect, employing evolution as a force for optimisation. Robots are set some specific task,

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 569–578, 2013.
© Springer-Verlag Berlin Heidelberg 2013

their performance is measured through some objective function and a, typically centralised, evolutionary algorithm optimises robot behaviour accordingly.

In our vision, evolution serves two purposes: on the one hand to allow robots to adapt to the environment and to behave so that they can operate at all. On the other hand, evolution is a force to promote task-performance, where we interpret 'task' in a broad sense: it is any user-defined preference with a measurable level of compliance. It can be a direct task, like collecting rubbish (measured by the amount of rubbish cleared), but also more indirect, like energy efficiency (measured by battery lifetime). Combining these two (seemingly) contradictory roles of evolution is a generic, fundamental challenge that to our knowledge has to date not been tackled successfully.

As Jones and Mataríc note [7], collectively tackling tasks also entails a division of work: if, for instance, the swarm has two (sub-)tasks, it may be possible that all robots perform both tasks or that part of the swarm focusses on one task and the other robots tackle the second task. Therefore, an algorithm that enables our vision of an adaptive collective of robots should combine the adaptive and optimising facets of evolution as well as promote a good division of labour.

This paper introduces the MONEE (**M**ulti-**O**bjective a**N**d open-**E**nded **E**volution) algorithm that combines the open-ended and task-driven aspects of evolution. It is inspired by the open-ended algorithms described in [2] and [19].

MONEE allows the robot collective to optimise their behaviour to suit multiple tasks while distributing the tasks over that collective. It extends the open-ended approach as found in MEDEA [2] with a currency-based system where an individual can earn as well as spend credits. Please note that our idea of open-endedness in this case entails the lack of an explicit fitness function, whether our system adheres to the more general sense of open-endedness where evolution does not converge has not yet been tested. The main idea is that earnings are based on task-performance, while spendings are related to reproduction. Individuals accumulate credits through task performance - the better a robot performs a task, the more credits it earns for that task. When an individual puts its genome forward as a potential parent, it also passes information on its earnings for each defined task as a parental investment. Section 3 provides more detail on our implementation of the MONEE algorithm.

This paper provides a proof-of-concept for the MONEE algorithm. We simulate a collective of e-puck robots that are set multiple tasks to ascertain:

- if the robots adapt their behaviour to suit the tasks;
- if all tasks are performed equally well;
- how the system reacts to changing tasks.

2 Related Work

Open-ended and task-related evolutionary robotics Evolutionary Robotics has been widely studied since the early 1990s [15]. Initially, research focussed on individual robots, but since then substantial effort has been directed at evolution in larger numbers of interacting autonomous robots in swarms [20], research

projects include for instance the Swarmanoid project [5]) or modular robots (e.g. M-tran[8]). In all these cases, evolution is used to achieve some fixed user-defined objective such as locomotion or explicit coordination.

Open-ended or objective-free evolution as well as self-replication have been studied in Artificial Life since Rasmussen's (1990) [16] and Ray's (1991) [17] work. Such research primarily investigates evolutionary dynamics in the absence of tasks, but as a result of implicit or environmental criteria that impact the ability to spread genomes through the population. This open-ended approach has gained interest from the evolutionary robotics community, for instance in Bianco and Nolfi's experiments with self-assembling organisms [1] and more recently in the mEDEA algorithm [2].

Open-ended approaches have been considered as a strategy to promote behavioural diversity in multi-objective settings by, for instance, Mouret and Doncieux [13]. Lehman and Stanley's novelty search [9] also embraces open-endedness to tackle elusive problems where a straightforward objective function leads to sub-optimal behaviour. These recent advances do define objective functions, though: the definition of novelty for Lehman and Stanley and the secondary objective for Mouret and Doncieux are ad-hoc, task-specific definitions of behavioural diversity that amount to tangential and creative redefinitions of the orginal objective function. Thus, such methods are not the completely objective-free approaches where survival and rate of procreation determine fitness rather than the other way around.

Parental Investment. When animals reproduce they invariably invest time and energy in their offspring, for which Trivers coined the phrase parental investment [21]: "Parental investment covers any cost that a parent incurs in looking after an offspring, be it in gamete production, gestation or care after birth". Parental investment has been investigated in biology, and theories have been proposed on the evolutionary origins of the differentiation between sexes. These theories have also been verified in ALife settings, including experiments with robots [10,22,19]. In artificial life parental investment is often used to give the offspring a starting value of (virtual) energy [11,12,3,18] and a parent's energy level is often linked to task performance (e.g., agents tasked with eating grass to gather energy [3]). While these approaches benefit the offspring of good individuals, none of these approaches use parental investment as a method for parent selection. Distributed on-line evolutionary systems such as Watson et al's embodied evolution [24] sometimes employ (virtual) energy as a currency to determine parent selection [24,25].

3 MONEE: Multi-Objective and Open-Ended Evolution

At its core, MONEE is an adaptation of the mEDEA algorithm described by Bredèche et al. [2] and Schwarzer's artificial sexuality algorithm [19].

The robot lifecycle in MONEE consists of two phases: life and rebirth. The robots have a limited, fixed, lifetime during which they perform their actions; moving about, foraging, et cetera. When their lifetime ends, they enter the rebirth

phase and become 'eggs': stationary receptacles for genomes that are transmitted by passing live robots. This rebirth phase also lasts a fixed amount of time, at the end of which the egg selects parents from the received genomes to create a new controller. The robot then reverts to the 'life' role with this new controller. Thus, robots (or rather, their controllers) can procreate by transmitting their genome to eggs, and the more eggs a robot inseminates, the more chances it has for procreation. Because the transmission of genomes is continuous and at close range (e.g. through infrared), the more a robot moves about the arena, the better its chances of producing offspring. This aspect of MONEE is clearly open-ended: there is no calculated performance measure that defines the chances of being selected as parent, there is no task. Only the environment dictates what robots may or may not become parents.

To add task-driven parent selection to this basic evolutionary process, we use parental investments. During their lifetime, robots amass credits by performing tasks. For instance, a robot could get one credit for every piece of ore it collects, one for successfully solving some puzzle, and so on. If multiple tasks are defined, the robots maintain separate counts for the credits awarded for each task, for instance one counter for the pieces of ore collected and another one for the number of puzzles solved. When a robot inseminates an egg, it passes these numbers along with the genome and the egg uses that information to select parents when it revives.

When a robot's egg phase finishes, it compares the parental investment for each genome it has received. To enable this comparison across task, the egg calculates an exchange rate between tasks. This ensures that genomes that invest in tasks for which few credits are found overall (presumably hard tasks) are not eclipsed by genomes that favour easier tasks. The pseudo-code in algorithm 1 details this auction scheme.[1]

The parental investments relate task performance to reproductive success: besides the open-ended goal of 'merely' transmitting genomes to eggs, robots must also become proficient at the defined tasks for these genomes to be selected. The more proficient a robot is at a task, the higher its chances of procreating. The comparison of investments across multiple tasks introduces an exchange rate between the earnings per task: the more common credits are for a particular task, the less their worth and vice versa. Thus, parent selection becomes a marketplace for skills and features that the user requires. Users can influence this economy to prioritise tasks, for instance by setting a premium on investments related to a particular task that the user deems more urgent than others. This system naturally caters for multi-objective approaches and allows the user to prioritise tasks in a straightforward manner.

[1] Note, that our implementation uses roulette wheel selection and only mutation only, so a single parent is selected. These are incidental design choices: MONEE does not preclude the use of other selection schemes and/or recombination operators.

Algorithm 1. Selecting a parent based on parental investments

```
for every defined task do                                   // total credits
    for every received genome do
        credits_task ← credits_task + genome.credits_task;
    end for
    credits_overall ← credits_overall + credits_task;
end for
for every defined task do                                   // exchange rate per task
    rate_task ← (credits_overall + num_tasks) / (credits_task + 1);
end for
for every received genome do                                // parental investment per genome
    for every defined task do
        genome.rating ← genome.rating + (genome.credits_task · rate_task);
    end for
end for
parent ← roulettewheel_selection(received genomes)          // select and mutate
child ← mutate(parent);
reactivate(child)                                           // revive
```

4 Experiments

To investigate the MONEE algorithm, we implement it in a scenario with simulated e-pucks in a simple 2D simulator.[2] This scenario places 100 robots in an arena roughly 330 robots wide, with a number of obstacles (depicted in the lower right of Fig. 1) and defines seven concurrent foraging tasks. Concurrent foraging is a variation of regular foraging where the arena is populated by multiple types of objects to be collected [7], rather than just a single resource. In our case, just as in [7], these objects are pucks of different colours, and the collection of each different colour is a different task.

We use seven differently coloured pucks, defining seven similar but different tasks. The colours are *SteelBlue*, *OrangeRed*, *LimeGreen*, *Indigo*, *SeaGreen*, *SandyBrown* and *Siena*. The pucks are spread in the environment according to a number of gaussian distributions as indicated in Fig. 1. As can be seen from the distributions, some colours (like *Indigo* and *SandyBrown*) are

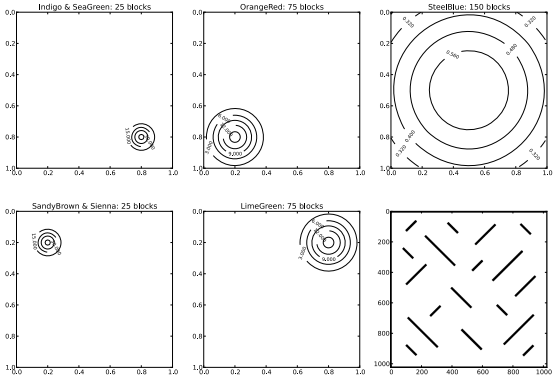


Fig. 1. Distribution of pucks of various colors and obstacles present in the arena

[2] RoboRobo, `https://code.google.com/p/roborobo/`

placed very compactly at the corners of the arena, which makes gathering those a more specialist proposal than for instance *SteelBlue* pucks, which can be found scattered across the entire arena. The number of pucks per colour varies between 25 and 150, also indicated in Fig. 1. When placing pucks we make sure that they do not overlap with existing pucks, robots or other obstacles.

Robots gather these pucks simply by driving over them, and as soon as a robot has gathered a puck it is immediately removed: the robots do not have to transport the puck to a particular region. A replacement puck of the same colour is then randomly placed in the arena, taking the appropriate distribution into account, to allow the experiment and evolution adequate time.

Each robot is controlled by a single-layer feed forward neural network which controls its left and right wheels. The inputs for the neural network are the robot's sensors: a robot has 8 sensors for each type of puck, as well as 8 sensors to detect environmental obstacles and other robots. The layout of these sensors is that of a standard e-puck's infrared sensors: four face forward, two to the sides and two face backwards. Because the robots have separate sensors for each type of puck and the network only has direct connections from input to output, the task of collecting each type of puck –although very similar– needs to be learned completely separately.

The robot's genome directly encodes the neural network's weights (8 types of sensor $\times$ 8 sensors $\times$ 2 outputs plus 2 bias connections plus 4 feedback (current speed and current rotation to either output) = 134 weights) as an array of reals.

As specified by the MONEE algorithm, the robots alternate between periods of explorative block gathering and motionless genome reception. To prevent synchronised cycles among the robots, we add a small random number to each robot's fixed lifetime. This forces desynchronised switching between life and rebirth even though our runs start with all robots perfectly in sync at the first time-step of their lifetime.

At the end of the egg phase offspring was created by selecting a parent from the received genomes according to the parental investment and mutating it using gaussian perturbation with a single, fixed mutation step size $\sigma = 1$.

To investigate the response of this system to dynamically changing tasks, we radically change the distribution of pucks during the runs: halfway through the run, at 1 million iterations, all distributions generate pucks of a single colour only: only *SteelBlue* in half the runs, only *Indigo* in the others (i.e., either a common or a rare task remains).

Due to time constraints, we were only able to conduct a limited number of runs. Therefore, it is not feasible to provide meaningful statistics on the exact level of performance. The data does suffice, however, to indicate the potential of the MONEE approach and to analyse some of the dynamics of a population of robots that use MONEE to adapt their behaviour.

5 Results and Analysis

Irrespective of the foraging tasks, the robots must cope with their environment to be able to procreate: they must at the very least develop controllers that drive

around the arena to inseminate eggs. Thus, the environment implies that the robots should move around. The robots should also avoid obstacles, even though this is not specified as an explicit task. If they do not, the time they spend trying to drive through an obstacle cannot be spent spreading their genome, limiting their chances of creating offspring. Therefore, we measure the number of collisions between robots and obstacles to gauge the level of adaptation to the environment. Figure 2 shows the total number of collisions for the whole collective over time. The number of collisions is aggregated over 1000 time steps. The number of collisions decreases with time. Even though the decrease is not spectacularly steep, it is a clear indication that controllers do adapt to the environment without any specifically set goal.

Of course, we did specify the foraging tasks for the robots. Figure 3 shows the number of pucks harvested against time, here too the number of pucks gathered is aggregated over 1000 time steps. Clearly, the robots do evolve effective foraging behaviour: the number of collected pucks per time unit steadily increases throughout the runs. Changing the environment so that only pucks of a single colour are generated in most cases actually leads to a slight increase in the total number of pucks gathered. This seems to indicate that the robots in those runs do not specialise in a particular task. Possibly, when only a single colour remains, they are not distracted by pucks of a different colour and they therefore forage more effectively.

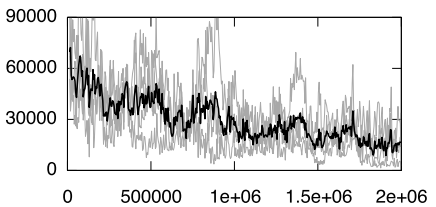


Fig. 2. Number of collisions over time. The grey lines denote individual runs, the black line shows the average over the four runs.

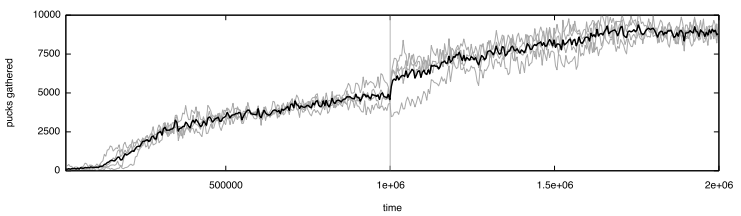
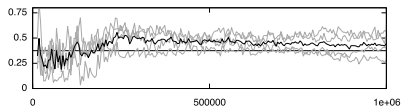


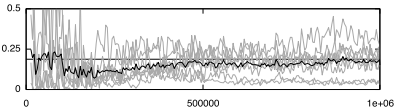
Fig. 3. Number of pucks gathered over time. The grey lines denote individual runs, the black line shows the average over the four runs. The vertical line indicates the moment where the task changes and all pucks become a single colour.

To asses the efficacy of MONEE's currency scheme to distribute tasks, we also ran our experiments with the exchange rate mechanism turned off. In this case, a genome's chance of selection is related purely to the number of pucks that it collected without any consideration for their colour. Therefore, genomes that encode harvesting behaviour for rare colours are at a disadvantage. Figure 4 compares the fraction of *SteelBlue*, *OrangeRed* and *LimeGreen* pucks out of

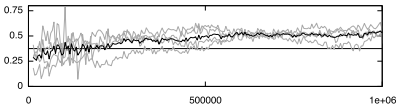
all gathered pucks with and without the exchange rate mechanism. The plots only show the first million time-steps because after that only a single colour remains as described above. With the exchange rate mechanism turned on (Figs. 4(a) and 4(b)). In both cases, the fraction of pucks gathered tends towards the actual fraction of pucks available: the trend in Fig. 4(a) decreases to the natural ratio of 0.375, indicating that the easiest task of collecting ubiquitous pucks is balanced with the harder task of collecting rarer pucks. Similarly, the fractions of *OrangeRed* and *LimeGreen* pucks in Fig. 4(b) slowly seem to increase to level off at the natural ratio of 0.1875. Figures 4(c) and 4(d) show a different picture. Without the exchange rate mechanism, the simple task is increasingly favoured as shown by the continuing rise of the *SteelBlue* fraction. This is at the expense of collecting rare colours, as indicated by the decreasing trend in Fig. 4(d).



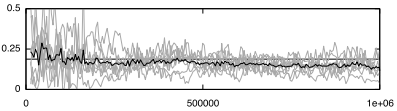
(a) *SteelBlue* fractions with auction



(b) *OrangeRed*, *LimeGreen* fractions with auction



(c) *SteelBlue* fractions without auction



(d) *OrangeRed*, *LimeGreen* fractions without auction

Fig. 4. Fraction of gathered pucks that are *SteelBlue* and *LimeGreen* for runs with and without the exchange rate auction mechanism turned on. Light grey plots for individual runs, black lines show the average over the runs. Horizontal black lines indicate the fraction of all pucks for the respective colours (0.375 and 0.1875).

To verify the decrease in collecting rare pucks, we ran a more extensive experiment. To isolate this claim more purely we used only 2 colours, red and blue, in a 1:3 ratio, i.e. 50 blue pucks and 150 red pucks. This amounts to a natural collection ratio of 0.25 for blue pucks. All other experimental parameters were kept the same as the previous experiments, except that we ran 64 repeats.

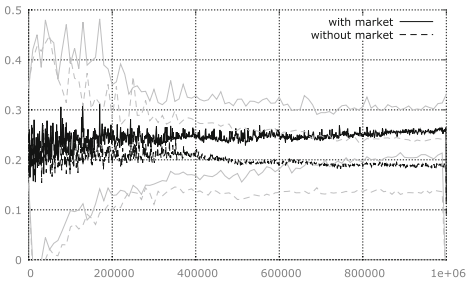


Fig. 5. Market fraction for verification runs. Grey lines indicate standard deviation.

Figure 5 shows the market fraction of blue pucks gathered, with and without market. As you can see the ratio for blue pucks gathered *with* market trends towards the natural ratio of 0.25, while the ratio for blue pucks gathered *without* market steadily drops over the course

of a run. The difference in ratios between experiments with and without market is significant (Kolmogorov-Smirnov: $p = 3.4867 \times 10^{-8}$, $D = 0.5156$).

6 Conclusions and Further Research

We have introduced the MONEE algorithm as a tool to combine the open-ended and task-driven facets of evolutionary robotics. As a proof of concept, we ran experiments where robots have to move about an obstacle-strewn arena while concurrently foraging seven types of puck. The robot controllers were laid out so that this amounts to having to learn seven distinct tasks.

MONEE allows the robots to learn to cope with their environment as shown by the decreasing frequency of collisions. It also drives task-driven adaptation: the robots become increasingly proficient at the gathering tasks and a momentous change where six of the tasks disappear has no ill effect on the collective task performance. The exchange rate mechanism allows effective division of the tasks over the collective without favouring easier tasks at the cost of harder ones.

We emphasise once again that these results are based on a very limited number of runs. Nonetheless, they provide a good indication of algorithm behaviour, enough at least to show that the MONEE algorithm opens a promising avenue of further research. We are of course planning to conduct further experiments to provide solid statistical foundations for the indications we show in this paper. Moreover, we are keenly interested in researching the intricacies of the economy that results from the exchange rate mechanism in the face of more dynamic changes in task composition as well as the results of enforcing some level of task specialisation.

Acknowledgements. This work is supported by: EU-IST-FET project 'SYMBRION', no. 216342. The authors would like to thank Nicolas Bredeche and our partners in the SYMBRION consortium for many inspirational discussions on the topics presented here.

References

1. Bianco, R., Nolfi, S.: Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce. Connection Science 4, 227–248 (2004)
2. Bredeche, N., Montanier, J.-M., Liu, W., Winfield, A.F.: Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. Mathematical and Computer Modelling of Dynamical Systems 18(1), 101–129 (2012)
3. Burtsev, M., Red'ko, V., Gusarev, R.: Model of Evolutionary Emergence of Purposeful Adaptive Behavior. The Role of Motivation. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, pp. 413–416. Springer, Heidelberg (2001)
4. DeJong, K.: Are genetic algorithms function optimizers? In: Männer, R., Manderick, B. (eds.) Proceedings of the 2nd Conference on Parallel Problem Solving from Nature, pp. 3–13. North-Holland, Amsterdam (1992)
5. Dorigo, M., et al.: Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. IEEE Robotics & Automation Magazine (2012) (in Press)
6. Eiben, A.E., Smith, J.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)

7. Jones, C., Mataric, M.: Adaptive division of labor in large-scale minimalist multi-robot systems. In: Proceedings. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2, pp. 1969–1974 (October 2003)
8. Kurokawa, H., Yoshida, E., Tomita, K., Kamimura, A., Murata, S., Kokaji, S.: Self-reconfigurable m-tran structures and walker generation. Robotics and Autonomous Systems 54(2), 142–149 (2006)
9. Lehman, J., Stanley, K.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation 19(2), 189–223 (2011)
10. Mascaro, S., Korb, K., Nicholson, A.: An alife investigation on the origins of dimorphic parental investments. In: Abbass, H.A., Bossomaier, T., Wiles, J. (eds.) Advances in Natural Computation, vol. 3, pp. 171–185 (2005)
11. Menczer, F., Belew, R.: Latent energy environments. In: Santa Fe Institute Studies In The Sciences of Complexity-Proceedings, vol. 26, pp. 191–210 (1996)
12. Menczer, F., Willuhn, W., Belew, R.: An endogenous fitness paradigm for adaptive information agents. In: CIKM Workshop on Intelligent Information Agents, Citeseer (1994)
13. Mouret, J.-B., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: an empirical study. Evolutionary Computation 20(1), 91–133 (2012)
14. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness functions in evolutionary robotics: A survey and analysis. Robotics and Autonomous Systems 57(4), 345–370 (2009)
15. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press, Cambridge (2000)
16. Rasmussen, S., Knudsen, C., Feldberg, R., Hindsholm, M.: The coreworld: Emergence and evolution of cooperative structures in a computational chemistry. Physica D: Nonlinear Phenomena 42(1), 111–134 (1990)
17. Ray, T.S.: Is it alive or is it GA? In: Belew, R., Booker, L. (eds.) Proceedings of the 4th International Conference on Genetic Algorithms, pp. 527–534. Morgan Kaufmann, San Francisco (1991)
18. Scheutz, M., Schermerhorn, P.: Predicting population dynamics and evolutionary trajectories based on performance evaluations in alife simulations. In: Beyer, H.-G., O'Reilly, U.-M. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005), pp. 35–42. ACM (2005)
19. Schwarzer, C., Hösler, C., Michiels, N.: Artificial sexuality and reproduction of robot organisms. In: Levi, P., Kernbach, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, pp. 384–403. Springer, Heidelberg (2010)
20. Trianni, V.: Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots, vol. 108. Springer (2008)
21. Trivers, R.: Parental investment and sexual selection. In: Campbell, B.G. (ed.) Sexual Selection and the Descent of Man. ch.7, pp. 136–179. Biological Laboratories, Harvard University (1972)
22. Ventrella, J.: Genepool: Exploring the interaction between natural selection and sexual selection. In: Artificial Life Models in Software, pp. 81–96 (2005)
23. Ward, M.: Virtual Organisms: The Startling World of Artificial Intelligence. Pan Books (2010)
24. Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and Autonomous Systems 39(1), 1–18 (2002)
25. Wischmann, S., Stamm, K., Wörgötter, F.: Embodied Evolution and Learning: The Neglected Timing of Maturation. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 284–293. Springer, Heidelberg (2007)

Virtual Spatiality in Agent Controllers: Encoding Compartmentalization

Jürgen Stradner[1], Heiko Hamann[1,2], Christopher S.F. Schwarzer[3], Nico K. Michiels[3], and Thomas Schmickl[1]

[1] Artificial Life Laboratory of the Department of Zoology, Karl-Franzens University Graz, Universitätsplatz 2, A-8010 Graz, Austria
[2] Department of Computer Science, University of Paderborn, Zukunftsmeile 1, 33102 Paderborn, Germany
[3] Institute for Evolution and Ecology, University of Tübingen, Auf der Morgenstelle 28, 72076 Tübingen, Germany

Abstract. Applying methods of artificial evolution to synthesize robot controllers for complex tasks is still a challenging endeavor. We report an approach which might have the potential to improve the performance of evolutionary algorithms in the context of evolutionary robotics. We apply a controller concept that is inspired by signaling networks found in nature. The implementation of spatial features is based on Voronoi diagrams that describe a compartmentalization of the agent's inner body. These compartments establish a virtual embodiment, including sensors and actuators, and influence the dynamics of virtual hormones. We report results for an exploring task and an object discrimination task. These results indicate that the controller, that determines the principle hormone dynamics, can successfully be evolved in parallel with the compartmentalizations, that determine the spatial features of the sensors, actuators, and hormones.

1 Introduction

In order to control a robot or a virtual agent some device capable of processing sensor input and generating actuator output is necessary. Besides the classical way of implementing agent controllers by means of software engineering or control theory, there is research pursuing an automatic synthesis of agent controllers. Examples are the fields of evolutionary robotics [11] and adaptive agents [4]. However, the (semi-) automatic synthesis of agent controllers is still challenging, especially in complex tasks, which is partially documented by the absence of complex benchmark tasks in the literature [10].

The standard approach for controller synthesis is based on loosely biologically inspired methods, such as Artificial Neural Networks (ANN) [11]. Additionally, also the vast variety of naturally evolved control systems is often reduced to the central nervous system of vertebrates. However, the relevance of different control systems in nature is, for example, constituted in unicellular organisms that often show non-trivial behavior (without a single nerve cell), such as *Paramecium* [5].

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 579–588, 2013.
© Springer-Verlag Berlin Heidelberg 2013

One key feature in such chemical driven information processing system is spatiality. Nature can be said to operate on spatiality beginning with the emergence of compartments in single cells. Biological systems evolved complex membranes to establish spatiality through compartments [1] which enable eucaryotic cells to perform multiple functions that would be mutually exclusive without local separations. These functions are implemented by chemical reactions which rely on compartmentalization [8]. Additionally, conditions are established which increase the efficiency of specific tasks. This results in a strong connection between spatiality and functionality.

The relation between function and spatial features is the focus of the work presented here. In previous works, a bio-inspired approach has been proposed for controlling robots ('Artificial Homeostatic Hormone System') guided by the examples of signaling networks in unicellular organisms [12]. This paper investigates spatial effects in those controllers and the potential to boost the performance of evolutionary algorithms. We use artificial evolution and spatial partitioning by Voronoi diagrams to investigate the interaction of evolving controllers and compartments concurrently. The rationale of this approach is that, on the one hand we create an Euclidean plane of controller features such as sensor or actuator IDs and on the other hand we also embed the computational process into this space by the compartment structure and the applied diffusion processes. The evolutionary algorithm operates within Euclidean space which simplifies, for example, the encoding of partitioning sensors into two groups just by adding a line into the plane. Hence the spatial encoding might offer shortcuts during optimization. Operating within continuous space also has the advantage that mutations typically have small effects which implements an efficient local search.

2 Artificial Homeostatic Hormone Systems

We use controllers based on Artificial Homeostatic Hormone Systems (AHHS) [13,15]. AHHS are a reaction-diffusion approach. Sensory stimuli are converted into virtual hormone concentrations that possibly interact with other hormones and finally control the actuators. Hormone concentrations diffuse through a virtual inner body of the agent, and decay over time. An AHHS controller consists of descriptions of hormones (production and decay rate, diffusion coefficient) and of descriptions of rules that define the dependency between sensor input and the corresponding changes in hormone concentrations, the interactions of hormones, and the mapping of hormone concentrations to actuator control values. A rule consists of four sub-rules: sensor, linear hormone-to-hormone, nonlinear hormone-to-hormone, and actuator. The parameters of hormones and rules are encoded as arrays of floating point numbers which represent the genome. They are subject to optimization of the controller's functionality by the evolutionary algorithm. We use only one hormone and up to 30 rules; for details, see [7].

The main focus here is on spatial properties of AHHS which are defined by compartments. The compartmentalization structures the virtual inner body of the controlled agent. Hormone concentrations diffuse from one compartment to neighboring compartments as described by

$$\frac{\Delta H_h^c}{\Delta t} = D_h \nabla^2 H_h^c(t) + C, \tag{1}$$

for a constant C, that includes all other details described above, whereas the diffusion process is discrete in the implementation. The main application area of AHHS is modular robotics [13] where a natural compartmentalization due to physically connected robot modules exists. Here, we focus on internal compartmentalizations within single-module agents.

3 Compartmentalization with Voronoi Diagrams

We propose an approach for evolving spatial features of controllers. This is in addition to previous works in which only functional features were adapted. We use Voronoi diagrams to describe compartmentalizations used by AHHS controllers and apply evolutionary operators similar to [14].

A Voronoi diagram is a decomposition of space, determined by the distances to a set of given points [16]. The following definition of Voronoi diagrams is based on Aurenhammer [2]. Let S be a set of points (called *sites*) in the plane. For two distinct sites $p, q \in S$ the dominance of p over q is defined by:

$$D(p,q) = \left\{ x \in \mathbb{R}^2 : |p - x| < |q - x| \right\}. \tag{2}$$

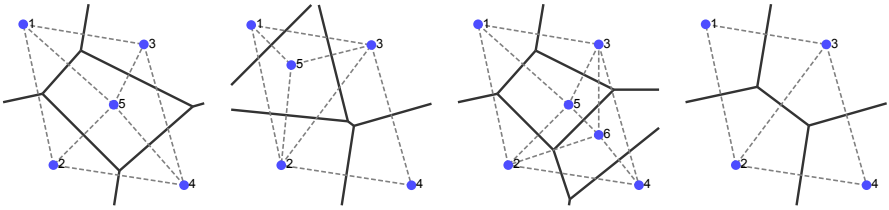
It is the subset of the plane being at least as close to p as to q. The region of a site $p \in S$ is the portion of the plane of all points dominated by p over the remaining sites as given by

$$R(p) = \bigcap_{q \in S \setminus \{p\}} D(p, q.) \tag{3}$$

All regions combined form a polygonal partition called Voronoi diagram. Informally, a region in a Voronoi diagram is all space that is closer to the site of that region than to any other sites. The Voronoi edge is the border between regions and contains points that are equally distant to both regions' sites.

Here, we use Voronoi diagrams in 2-d space to describe the compartmentalization of AHHS. Each Voronoi region corresponds to a compartment of the AHHS which holds hormone concentrations. The genome for the AHHS is extended by a Voronoi genome that consists of a set of points in 2-d coordinates which create the sites of a Voronoi diagram. We directly compute the Delaunay graph of the Voronoi sites and simulate the hormone diffusion along its connections.

Each sensor and actuator is associated with an *anchor point* in the plane, which is its virtual position. Hence, artificial evolution can be applied in two ways: On the one hand, it is an option to evolve the virtual position of the sensors and actors of the agent by mutating the *anchor points*. On the other hand, the compartment structure can be evolved. Some mutations might be silent due to their effect being too small for a change in the Delaunay graph. Others, concerning the compartment structure, are summarized in Fig. 1. Fig. 1(a) shows the original Voronoi genome to which the following three operators are applied:



(a) Original Voronoi diagram. (b) A shift, causing a connection to flip. (c) Point insertion, breaking up an existing connection. (d) Point deletion, one bridging connection is created.

Fig. 1. The original Voronoi diagram shown in (a) is mutated by point shifting (b), point insertion (c), and point deletion (d). Borders of Voronoi regions are solid lines. Dashed lines are region neighborships, which is the Delaunay graph.

Point Shifting: This mutation translates a site $p = (p_1, p_2)$ to $p' = (p_1 \pm \Delta_1, p_2 \pm \Delta_2)$ for random variables Δ_1 and Δ_2. Shifting the point does trigger a connection flip. For example in Fig. 1(b), instead of the connection between points 4 and 5, there is now one between 2 and 3.

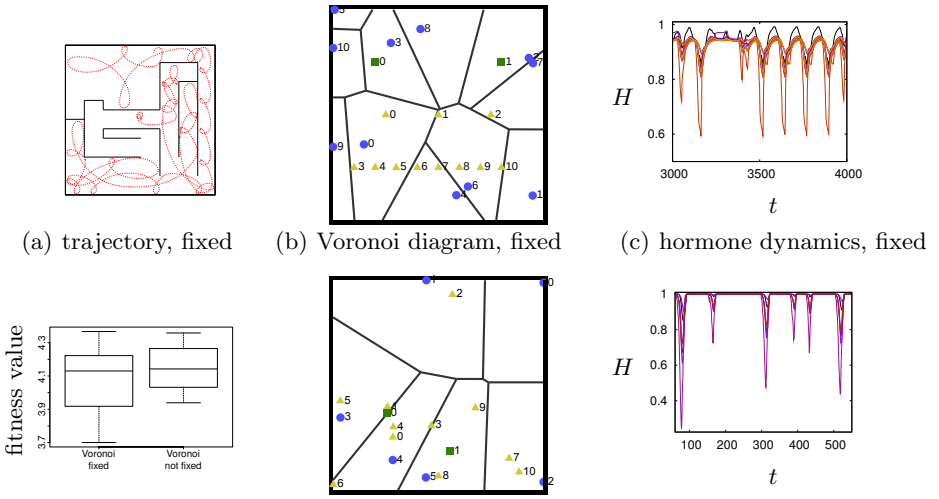
Point Insertion: When a site p with random coordinates is inserted $S' = S \cup \{p\}$, a new compartment is put into the neighborhood of cells. A point insertion can cause a change to the existing neighborhood connections, as seen in Fig. 1(c) where the new point 6 breaks up the neighborship between points 4 and 5 by being put in between.

Point Deletion: Removing a site p from the Voronoi genome $S' = S \setminus \{p\}$ causes its region in the Voronoi diagram to disappear and its space being taken over by its former neighbors. This can lead to new connections as regions that were separated before might be neighbors now. For example the regions of points 2 and 3 in Fig. 1(d) are neighbors after removing point 5.

We implemented the entire controller approach which is evolvable for functionality and spatiality into agents which had to perform two tasks of different complexity. In the first task, evolution operates on both Voronoi regions and the anchor points of sensors and actuators. In the second task, the evolution of compartmentalizations while keeping anchor points fixed is compared to the evolution of only space-independent controller features (rules and hormones). In the following these tasks are described and the results are presented.

4 Exploring Task

In this scenario, the task is to explore a maze consisting of walls, see Fig. 2(a). The robot depends on its proximity sensors. The arena is divided in patches to measure the performance of the controller. Fitness is increased by visiting new patches. The maximally achievable fitness within 5000 time steps is about 5.8. A simple wall-following behavior is beneficial, but twisting trajectories might



(a) trajectory, fixed (b) Voronoi diagram, fixed (c) hormone dynamics, fixed

(d) comparison (e) Voronoi diagram, not fixed (f) horm. dynamics, not fixed

Fig. 2. Analysis of evolved controllers: (a), (b) and (c) fixed sensor/actuator anchor points, fitness is 4.24; (e), (f) mutated sensor/actuator anchor points. (a) trajectory. (b), (e) Voronoi diagram, anchor points of actuators (squares), anchor points of sensors (triangles), Voronoi sites (circles). (c), (f) hormone dynamics for all compartments. (d) comparison of fixed and not fixed sensor and actuator anchor points.

allow for traversing even more patches. The motivation to investigate this task is that the connection between the required behavior and the spatial organization of the controller is evident and we want to test whether our approach discovers this solution. A proximity has to be created between either the left actuator and the sensors pointing to the left half or the right actuator and the right-hand sensors plus an inhibitory effect of high sensor input (i.e., close wall) on the actuator. We initialize with random rules, mutate, and recombine them. We compare two variants of evolving spatiality. The first variant is based on predefined anchor points of sensors and actuators in the Voronoi plane and keeping them fixed (called 'fixed'). The second variant is to initialize these anchor points randomly and then to mutate their positions (called 'not fixed'). The population size is 200 and the number of generations is 300.

An example of an agent's behavior controlled by an AHHS with fixed anchor points presented as a trajectory is shown in Fig. 2(a). The performance comparison due to fitness between controllers using fixed anchor points in the Voronoi plane and controllers with evolved anchor points is shown in Fig. 2(d). There is no significant difference. Hence, a predefined setting is not necessary and no previous knowledge is needed in this task and effective behaviors can be evolved.

In the following, we analyze two evolved controllers: one with fixed anchor points and one with evolved anchor points. In both approaches effective controllers were evolved. We start with the controller that was evolved with fixed

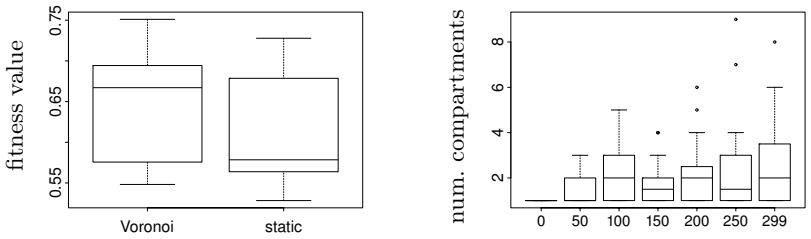
anchor points. The two anchor points of the actuators were placed in the upper third of the Voronoi plane, see squares in Fig. 2(b).

The anchor points of the proximity sensors are placed in one line in the lower third: triangles S_3 through S_{10} (sensors S_0, S_1, S_2 are not in use here). The evolved compartmentalization shown in Fig. 2(b) is almost symmetric and effective because it combines the sensors S_3, S_4, and S_5, that perceive the right side, in compartment 0 which is neighboring compartment 3 which contains the right actuator A_0. The sensors S_8, S_9, and S_{10}, that perceive the left side, are combined in compartment 6 which is neighboring compartment 2 which contains the left actuator. Summarizing the functionality of the evolved rules, the proximity sensors to the left control the left actuator inhibitorily (i.e., left wheel slows down when approaching a wall) while the right actuator always goes full speed. The trajectory contains left turns only. The hormone concentration is close to 1 in situations the agent is far from a wall and it is much lower when approaching a wall. In Fig. 2(c) drops in the hormone concentrations of all compartments are seen which are associated with approaches to walls.

The main feature of the compartmentalization for the case of evolving the anchor points as well, see Fig. 2(e), is that proximity sensor S_5 (perceiving walls to the right) and the right actuator A_0 are both in compartment 3. They are separated from the compartment containing the left actuator A_1 by one other compartment. Similarly the proximity sensors at the left side of the agent (S_7, S_8, S_9), which could disturb the sensor input from S_5, are also separated by at least one other compartment. The evolved rules are inhibitory again. The actuation of the wheels is maximal and the hormone concentration is 1 when walls are far, see Fig. 2(f). Once a wall is perceived on sensor S_5 the left actuator is not only slowed down but even turned to backward motion. This allows for a small turning radius in the right turns (trajectory not shown).

5 Object Discrimination Task

This scenario is an active categorical perception task, or short: object discrimination task, following [3]. Related works are [6,9]. The robot moves in 1-d (left and right on an interval $[0, 1]$) while objects are falling down from top. The objects are either circles with different diameters or lines with different lengths. The robot has to move as close to circles' impact positions as possible while it has to avoid those of lines. It has seven proximity sensors pointing to the top, distributed over an angle of $\pi/6$. At the beginning of an evaluation the robot is placed at $(0.5, 0)$. The objects' initial position is $(x, 2)$ with uniformly randomly distributed $x \in [0.1, 0.9]$ and their lengths/diameters are uniformly randomly distributed over $[0.01, 0.19]$. The robot moves only horizontally with a speed up to ± 0.05 units per step. The objects move only vertically with constant speed of -0.03 units per step. Hence, they touch the ground after 67 steps. We have a population of 100 and 300 generations. The fitness of one evaluation is the distance d_{line} between robot and line or $1 - d_{\text{circle}}$ in the case of a circle. Controller's total fitness is an average of 60 evaluations. The motivation for this task



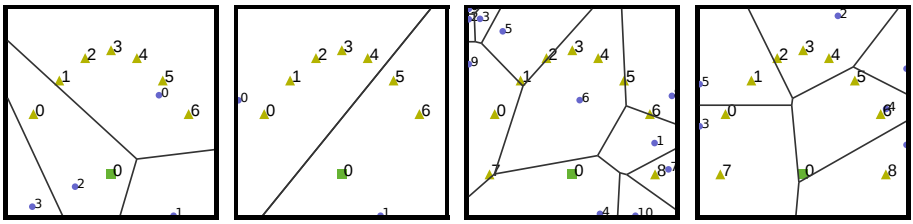
(a) comparison of best evolved controllers with and without evolving compartments based on Voronoi diagrams, $n = 40$ samples each

(b) number of compartments of the best evolved controllers, $n = 20$ samples

Fig. 3. Object discrimination task, comparison of best fitness between evolving compartments and not evolving them and increase of compartments for the former case

is that the connection between required behavior and spatial controller features is not evident because it seems beneficial to consider the input of all sensors equally. Hence, we want to investigate whether a spatial approach can still be effective in this kind of tasks. We focus on evolving a) the functional features of an AHHS controller only (hormone and rule genomes), in comparison to b) additionally evolving compartmentalization. Given appropriate parameter settings of the AHHS, satisfying solutions for this task can be evolved with and without evolving compartmentalizations. In this work, we are however interested in the differences between these two approaches. Therefore, we restrict the settings to minimal AHHS controllers that are barely able to solve the task. That way we can find substantial differences in these two methods based on the maximally achieved fitness. We allow only one hormone and four rules for the AHHS controllers and we initialize it with one compartment. In a first set of 40 runs we allowed the evolution of compartmentalizations, hormones, and rules. In the second set of 40 runs we did not evolve compartmentalizations but only hormones and rules. The results are shown in Fig. 3(a). The 'Voronoi' approach turns out to be significantly better (Wilcoxon rank sum test, $p = 0.03947$). As stated above the system is initialized with one compartment. When the compartmentalization is allowed to be evolved, new compartments are generated by adding new Voronoi points (existent points are mutated). This use of increase is shown in Fig. 3(b). Since there is no explicit cost imposed on the number of compartments the total number is expected to increase; also because new compartments at the margins have small effects on the robot behavior.

About 65% of runs return a best controller with several compartments. In Figs. 4(a and b) we give two typical representatives of these evolved compartmentalizations. There seems to be a tendency to include the actuator in a compartment with some of the outside sensors. We can only speculate about the cause, it seems that it is relevant to react especially to objects that are not directly above the robot. The analysis of the evolved controllers is difficult in spite of the minimality of the AHHS controllers. In the following we analyze a

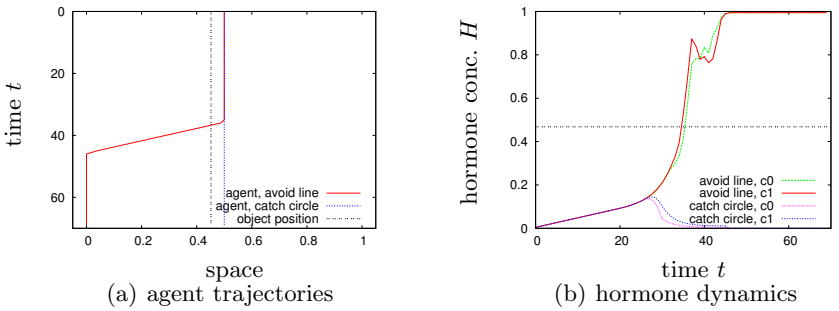


(a) standard task (b) standard task (c) noisy sensors (d) noisy sensors

Fig. 4. Typical evolved compartments of the best controllers for the standard object discrimination task and with two additional, disturbing noisy sensors (S_7 and S_8)

behavior that is not prominent in its performance but in its simplicity of analysis. This controller is able to catch most of the circles that appear close to the middle and is good in avoiding lines appearing in the right half of the arena. Rule 1 mainly generates small positive actuator input for a huge hormone concentration interval $H > 0.1$. Rule 2 reduces hormone intensively for sensor input $S > 0.468$ and additionally generates big negative actuator input for $H > 0.468$. Rule 3 is mostly ineffective because it only produces small actuator input for $H \approx 0.5$ which occurs only shortly in typical runs of this controller. Finally, rule 4 is ineffective because its trigger window width is set to 0. The resulting behavior is seen in Fig. 5. This controller pursues a simple strategy of staying centered for big sensor input ($S > 0.468$) and moving to the left end of the arena if the sensor input was not big enough until $t \approx 35$. Due to the extension in the y-dimension of circles this seems to be a possible strategy which, however, has a quite high ratio of mismatches. In Fig. 5(b) the hormone dynamics for two situations, a circle or a line at position 0.45, is given. The horizontal line gives the threshold of rule 2 (0.468). In case of the circle, sensor input is big ($S > 0.468$) before $t = 35$ and the hormone concentration in both compartments is reduced to zero which keeps the robot moving very slowly (due to rule 1). In case of the line, sensor input stays smaller ($S < 0.468$) before $t = 35$, the hormone concentration increases, when $H > 0.468$ the robot starts to move fast to the left due to rule 2.

In order to document this system's capabilities of separating sensors and actuators from each other by compartmentalizations we have done additional test runs. For this we have extended the object discrimination task by adding two more sensors: S_7 and S_8. We have placed them in the lower third of the Voronoi diagram to the left and the right of the actuator. These two new sensors, however, are of no purpose because they only exhibit random uniform noise over the full interval $[0, 1]$. When evolving controllers for this configuration we would expect that it is beneficial to separate sensors S_7 and S_8 from the actuator by introducing appropriate compartments. Indeed, this was observed in several of our test runs, see Figs. 4(c and d). In the case of the example shown in Fig. 4(c) most of the compartments containing sensors are directly neighboring the actuator's compartments. Sensor S_7 is as close to the actuator as several purposeful sensors. Still, by combining sensors S_2 through S_5 in one compartment might



(a) agent trajectories (b) hormone dynamics

Fig. 5. Agent trajectories and hormone dynamics of the analyzed controller for the two cases (avoid line and catch circle)

help to have a bigger effect on the actuator than sensor S_7. Sensor S_8 is separated effectively with a distance of two compartments. The example shown in Fig. 4(d) is better in separating the noisy sensors by pooling two purposeful sensors (S_5 and S_6) with the actuator in one compartment. In addition, a low diffusion rate minimizes the disturbance by the noisy sensors.

6 Conclusion

We have presented a concept of artificial spatiality for robot controllers inspired by compartmentalizations in natural cells. In the first task the performance of two aspects of spatiality were compared: the topology of the compartments vs. the virtual positions of the agent's sensors and actuators. This is indicated by evolving useful, symmetric compartments in case of fixed sensor/actuator anchor points, see Fig. 2(b) and especially in the case of evolved sensor/actuator anchor points where functionally associated actuators and sensors were put close together and separated from possibly perturbing sensors, see Fig. 2(e).

This is comparable to natural systems that learn while the position of their actuators change, for example, due to growing processes. In contrast to the optimization process in living systems, in artificial systems the evolutionary change of the inner morphology (i.e., the controller topology) can serve as a seed to optimize functionality. The combined evolution of morphology and function allows for alternative pathways through search space to desired agent behaviors that seem to improve the overall performance of the evolutionary approach. For example, the separation of sensors into two groups is easily achievable by the Voronoi approach (e.g., as seen in Fig. 4(b)) while it is more difficult to be evolved in a more abstract search space based (e.g., on sensor IDs). An additional advantage of evolving compartmentalizations is the easy visualization and, thus, the possibility of understanding the controller's mode of operation intuitively compared to sometimes complex networks of causality in the evolved logic of AHHS rules.

The idea to apply spatial features to information processing is rather new in artificial agents. Our approach is based on concepts which are known from

nature and well tested by natural evolution. Here we were able to show promising results based on a spatial approach in autonomous agents. In future work, we will compare our approach to standard approaches of evolutionary robotics and investigate the advantage of spatiality in controllers for complex tasks.

Acknowledgments. This work is supported by: EU-IST-FET project 'SYMBRION', no. 216342; EU-ICT project 'REPLICATOR', no. 216240.

References

1. Alberts, B.: Molecular biology of the cell. Garland Pub. (1989)
2. Aurenhammer, F.: Voronoi diagrams — a survey of a fundamental geometric data structure. ACM Computing Surveys 23(3), 345–405 (1991)
3. Beer, R.D.: The dynamics of active categorical perception in an evolved model agent. Adaptive Behavior 11(4), 209–243 (2003)
4. Beer, R.D., Gallagher, J.C.: Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior 1(1), 91–122 (1992)
5. Bray, D.: Wetware: A Computer in Every Living Cell. Yale University Press (2009)
6. Dale, K., Husbands, P.: The evolution of reaction-diffusion controllers for minimally cognitive agents. Artificial Life 16(1), 1–19 (2010)
7. Hamann, H., Schmickl, T., Crailsheim, K.: A hormone-based controller for evaluation-minimal evolution in decentrally controlled systems. Artificial Life 18(2), 165–198 (2012)
8. Lodish, H., Berk, A., Zipursky, L.S., Matsudaira, P., Baltimore, D., Darnell, J.E.: Molecular Cell Biology, 5th edn. W.H. Freeman and Company (2003)
9. Moioli, R., Vargas, P.A., Husbands, P.: Exploring the kuramoto model of coupled oscillators in minimally cognitive evolutionary robotics tasks. In: WCCI 2010 IEEE World Congress on Computational Intelligence - CEC IEEE, pp. 2483–2490 (2010)
10. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness functions in evolutionary robotics: A survey and analysis. Robotics and Autonomous Systems 57, 345–370 (2009)
11. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press (2000)
12. Schmickl, T., Hamann, H., Crailsheim, K.: Modelling a hormone-inspired controller for individual- and multi-modular robotic systems. Mathematical and Computer Modelling of Dynamical Systems 17(3), 221–242 (2011)
13. Schmickl, T., Hamann, H., Stradner, J., Crailsheim, K.: Hormone-based control for multi-modular robotics. In: Levi, P., et al. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, pp. 240–263. Springer (2010)
14. Schoenauer, M., Kallel, L., Jouve, F.: Mechanical inclusions identification by evolutionary computation (1996)
15. Stradner, J., Hamann, H., Schmickl, T., Crailsheim, K.: Analysis and implementation of an artificial homeostatic hormone system: A first case study in robotic hardware. In: The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), pp. 595–600. IEEE Press (2009)
16. Voronoi, G.: Nouvelles applications des paramétres continus à la théorie des formes quadratiques. Journal für Reine und Angewandte Mathematik 133, 97–178 (1907)

Evolving Counter-Propagation Neuro-controllers for Multi-objective Robot Navigation

Amiram Moshaiov and Michael Zadok

Faculty of Engineering, Tel-Aviv University, Israel
moshaiov@eng.tau.ac.il, michaelzadok@walla.co.il

Abstract. This study follows a recent investigation on evolutionary training of counter-propagation neural-networks for multi-objective robot navigation in various environments. Here, in contrast to the original study, the training of the counter-propagation networks is done using an improved two-phase algorithm to achieve tuned weights for both classification of inputs and the control function. The proposed improvement concerns the crossover operation among the networks, which requires special attention due to the classification layer. The numerical simulations, which are reported here, suggest that both the current and original algorithms are superior to the classical approach of using a feed-forward network. It is also observed that the current version has better convergence properties as compared with the original one.

1 Introduction

In many Evolutionary Robotics (ER) studies either an Elman or a simple Feed-Forward Network (FFN) is used (e.g., [2]). Here, following our recent investigation in [9], we deviate from that trend by using a Counter-Propagation Network (CPN). CPN includes a self-organizing (instar) network of Kohonen [7] as a first layer and a Grossberg's outstar net [3] as the second one. Such an approach differs from the common methods, as it involves not only training of a mapping from sensed information to actions, but also organization of the sensed information into classes based on a similarity measure. As noted in [9], CPN has not been used previously in ER studies (to the best of our knowledge). To employ CPN in an evolutionary context we devised a special training approach [9]. We have postulated, in [9], that the use of a CPN, backed with multi-objective search, may help reducing the need for a semi-manual approach. To back up this conviction, reference [9] provides a demonstration that the proposed CPN approach deals well with an environment, which was adopted from the study on a semi-manual approach in [4]. It has been concluded in [9] that the trained CPN-based Neuro-Controllers (NCs) may adapt well to a navigation problem and an environment, which differ from the trained ones. While presenting the potential of the use of a CPN approach to ER, our previous study in [9] includes no comparison of the proposed CPN approach with the classical approach of using feed-forward networks.

This paper follows the approach in [9]. It provides a modified algorithm, which is based on the observation that crossover among CPNs may have a detrimental effect

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 589–598, 2013.
© Springer-Verlag Berlin Heidelberg 2013

on the results. Next, the results obtained by the new algorithm are compared with those achieved using the algorithm of [9]. Moreover, an additional comparison is carried out with results obtained using a feed-forward approach. We restrict the current comparisons to the trained environment; the reader is referred to the study in [9], which elaborates on the generality and adaptation qualities of the controllers obtained by the proposed approach.

The rest of this paper is organized as follows. Section 2 provides some references and details relevant to the foundations of our approach. It is followed by a methodology, in section 3, which describes the details of the current search approach for the Pareto-optimal CPN-NCs. The results of the numerical simulations are presented in section 4. Finally, the conclusions from this study are provided in section 5.

2 Background

In the last decade, with the availability of Multi-Objective Evolutionary Algorithms (MOEAs), e.g. [1], several Multi-Objective ER (MO-ER) studies employed MOEAs to obtain Pareto-optimal NCs based on contradicting objectives (see a review in [8]). Pareto-based search deals with finding Pareto-optimal set, or its numerical approximation, using dominance relation. A Pareto-optimal set includes non-dominated solutions from the feasible search space given no a-priori preferences on a finite set of objectives which are contradicting. The Pareto-front is the set of performance vectors in objective space of all solutions of the Pareto-optimal set.

The usefulness of diversity, as obtained by a Pareto-optimality approach, has been recently demonstrated, in [10] and [6], for the bootstrap problem that is common in single objective ER. Such studies suggest that reaching diverse behaviors for one problem may produce useful (initial) solutions for another problem. The motivation to use a multi-objective approach is two-fold. First, as in [10] and [6], it provides diversity, which may help coping with numerical problems. The second, as in [8] and similar studies, it provides useful controllers for different scenarios. In particular, similar to [8], we use the trade-off between safety and target-attraction to produce a diversity of controllers, with remarkable different behaviors. To obtain diverse solutions we employ NSGA-II, [1], a well known MOEA, as the evolutionary search mechanism. Due to the permutation problems the use of a genetic algorithm is not recommended for the evolution of neural-networks [11]. Hence, as pointed out in [8], NSGA-II may not be the most optimal search algorithm for NCs, and a modified version, as used in [8], may be better. Yet NSGA-II proved to be useful for our current demonstration purposes.

The original idea of mixing Kohonen and Grossberg layers is attributed to Hecht-Nielsen [5]. While a promising concept, their use is not as common as that of FFNs. With increasing interest in cognitive robotics, the type of training should shift, from simple behavior-based mappings of sensors to actuators, to more complex approaches. CPNs are one such possibility, which has not been investigated in the context of ER. The advantage of using CPNs is that, once trained, they provide knowledge about the environment in the form of input classes. In regular training of CPNs there are two

phases. The first is to cluster the inputs, and the second is to create a mapping by the use of a supervised approach. The unsupervised learning, in the Kohonen self organizing layer, is commonly based on neighborhood functions. This means that weight adjustments are done not only for the winner neuron but also to its neighboring units [7]. Due to the lack of a supervisor in ER, and due to the learning by interactions with the environment, there is a need to re-examine existing CPNs learning algorithms. In particular, there is a need to investigate various alternatives to evolutionary training of CPNs, and to compare it with other approaches. In [9], we have suggested one possible pseudo-code that can be used either with a single objective ER or for MO-ER. As shown in [9], the obtained solutions support coping with shifts from one environment/problem to the other. Here we modify the algorithm of [9] to improve the results.

3 Methodology

3.1 Simulated Robot

To study the possible use of CPN in the context of ER, we followed the simulation details of the robot kinematics, and multi-objectives, of [8], while using environments as in [4]. As in [8] and in [9], we concentrate on producing a simulation-based population of NCs. Such solutions may be considered as candidates for use, with adaptation, in actual testing, which is likely to be required for coping with un-modeled aspects of the simulation.

The robot model is based on the miniature 5.5 cm diameter Khepera robot, as in [2] and in [8], with the following modifications. A total of 16 simulated sensors is used. Eight simulated IR sensors identify obstacles (walls) and the others identify targets. The sensors are located, as pairs of an obstacle and a target sensor, at eight locations as shown in figure 1. All simulated IR sensors have the same characteristics. The max range of any IR sensor is 5cm and its span angle is 6^0 (as shown for one of them).

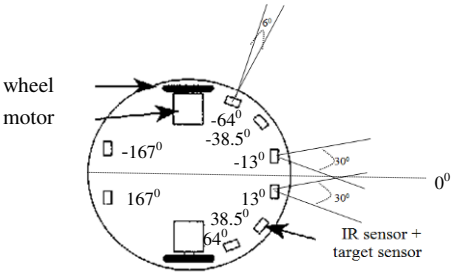


Fig. 1. Robot and sensors

The target sensors have a broader span of 30^0 (shown for the front sensors). The max range of the target sensor is simulated to be 100cm, which ensures target sensing anywhere in the maze. The sensors do not have a "blind range," and their output range is [0, 1]. The zero represents an object found at the max range, and the one is for the case when the sensed object is at the robot periphery. In the current implementation no noise is added to the simulated sensors, which is left for future research.

The simulated robot model converts motor commands on rotational speeds of the robot wheels from the outputs of the NNs into simulated robot motions. The range of the wheel speeds is scaled into the range [-0.5, 0.5]. The wheels radius is taken as 1cm. The time-step of the simulation is set to 5sec, and the robot moves 2.5cm per step at maximum speed.

3.2 Trained Environment

The trained environment, which is depicted in figures 2, is based on [4]. According to [4], the trained environment concerns nine different types of robot situations such as a wide corridor, a narrow corridor, the need to turn right/left, pass freely without walls, etc. In contrast to [4], our approach does not use separated simple environments for a semi-manual training. Rather, we use the complex environment directly for a non-manual evolutionary training.

We use several targets that the robot should reach. The targets are designed to specific positions including: (90.0, 6.0), (67.5, 6.0), (60.0, 15.0), (60.0, 42.0), (90.0, 45.0), (70.5, 51.0), (60.0, 19.5), (90.0, 15.0), (75.0, 30.0), (51.0, 15.0), (45.0, 40.5), (30.0, 55.5), (3.0, 45.0). These are shown, using dots, in figure 2. Spreading the targets aims to create an evolutionary pressure towards the different regions of the maze. In addition, we allocated a place in the maze with no targets. This supports simulating areas that are less desirable to be reached. For training we have used four different robot start-points located at (95, 5), (95, 45), (15, 5), (15, 45). In the two left points the robot is facing towards the right and vice versa.

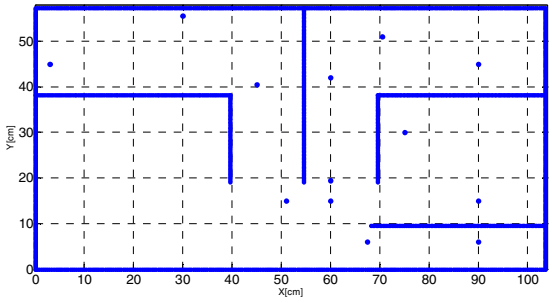


Fig. 2. Trained environment

3.3 Simulated Neuro-controllers

The simulated NCs maps sensors' information (input) into appropriate motor commands (output). The simulated CPN has two layers: the input layer connects the 16 sensors to a hidden layer. The hidden layer has 9 neurons that connect to two neurons in the output layer (motor commands). The reason of using a hidden layer of 9 neurons is that we try to compare it with the MNN (Modular-Neural-Network) of [4]. The 9 neurons follow the 9 classifications used in [4]. We use a 163 length vector to define each NC. This is based on: 16 weight inputs multiply by 9 neurons in the hidden layer + 9 weight neurons multiply by 2 outputs neurons. An additional weight is used to determine the slope of the sigmoid for the activation functions. No bias weights are used. In the current implementation of the CPN it has been observed that there is no need to adjust the weights of neighboring neurons, hence only the weights of the winner are adjusted.

3.4 Objective Functions

Two fitness functions are used (marked as F1, F2). The details are similar to [8]. The first function F1, which is based on [2], aims at fast and straight motions with obstacle avoidance without any specific destination. F1 is given as follows:

$$F_1 = \frac{\sum_{j=1}^{act\_step} V_j(1-\sqrt{\Delta v})_j(1-I)_j}{\max\_step} \quad \begin{matrix} 0 \le V \le 1 \\ 0 \le \Delta v \le 1 \\ 0 \le I \le 1 \end{matrix}$$

Where:

- V is the absolute value of the sum of the rotational speeds of wheels. V is high when the robot is moving fast (forward or backward).
- Δv is the absolute value of the difference between the wheel speeds. $1-\sqrt{\Delta v}$ is high when the robot is moving straight without making any turn during the step.
- I is the normalized activation value of the sensor with the highest value. I is high if the sensors perceive an obstacle.

F1 is calculated as an average over the maximum allowable number of steps of the accumulated score. The accumulation, however, is over the actual number of steps which are performed over a run of any particular NC. The function can have any value between 0 and 1, with the aim to be maximal. The second objective, F2, concerns reaching targets (e.g., food-targets). F2 is defined as follows:

$$F_2 = \sum_{j=1}^{act\_step} \frac{f_2}{\max\_step}; \quad f_2 = \begin{cases} H & if\ hit\ target \\ \frac{1}{1+d} & else \end{cases}$$

Where:

- d is the distance from our robot to the nearest target among the remaining targets at the current step.
- H is the score that the robot gets when it reaches a target. Here H is set to 50.

Similar to F1, F2 is based on averaging of performances over the maximum allowable number of steps of the process, and summing over the actual number of steps. This reduces scores to non-moving robots at the training phase. Once a target is touched by the robot, it is eliminated (consumed). After the robot finishes touching all targets, they re-appear. Then the process of reaching targets continues as long as the robot does not reach a maximum allowable number of steps.

3.5 Evolving CPN

As explained in [9], training a CPN requires special care due to the existence of two separated training issues. To achieve the required learning we have proposed a two-phase evolutionary search (in [9]), as depicted in figure 3, where the 1st phase involves the left side and vice versa.

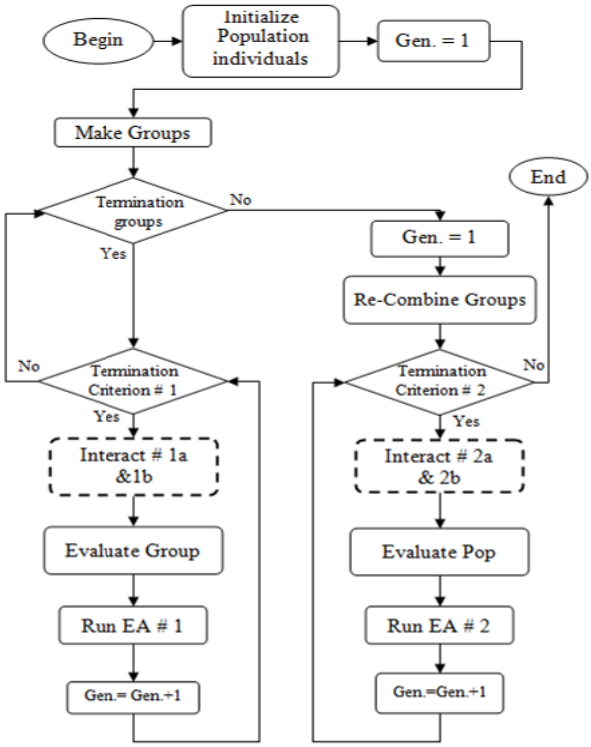


Fig. 3. Pseudo-code Description

The primary difference between the 1st phase and the 2nd one is that in the 1st phase the population is divided into groups, whereas in the 2nd one the groups are merged into one population. In the 1st phase each group has a different starting point

in the environment; this separation supports learning, at an early stage, the various classes of inputs that the entire environment contains. The proposed procedure can be used for both single and multi-objective problems.

Referring to figure 3, the main difference from the pseudo-code of [9] is in the cross-over operation of the 2nd phase, as further described below. At the beginning of the 1st phase, a random population is initialized with N individuals. Each individual is a CPN-based NC with a fixed structure, were both weights of the Kohonen layer and weights of the Grossberg layer are sought. Four groups are used in the current study, corresponding to the four start-points of the trained environment, as described in the previous section. During the 1st phase individuals evolve only within the group. In our study we set N=56 to be divided into four groups of 14 individuals each. A *"Termination Group"* criterion is used to terminate the 1st phase of the algorithm after the completion of the evolution of the four groups. A *Termination Criterion #1* is used to terminate the evolution of each group. In the 1st phase a maximal number of generations is used (50 generations per group). At each generation each individual performs two consecutive sequences of interactions with the environment, both starting at the corresponding start-point of its group. The purpose of the 1st sequence (Interact # 1a) is to update the weights of the Kohonen layer, whereas the 2nd sequence supports updating the Grosberg layer. In the 1st sequence the updates are done at each step of the sequence. The final Kohonen updates form Interact # 1a are used for the 2nd sequence, which aims to obtain the performances F1 and F2 of the individuals based on their updated version of the Kohonen layer. In the 2nd sequence no weight update is done during the sequence of interactions with the environment. Each robot finishes the interaction (#1a and #1b) either due to an obstacle or by reaching a pre-defined number of steps (200 steps in the current implementation). Following the interactions each of the groups' individuals is evaluated using F1 and F2 based on the accumulated scoring during Interact # 1b. In the current implementation the search in EA # 1 is for the Pareto-optimal set and front using NSGA-II based on [1]. The results include offspring population (of the group) to be evaluated in the next group generation of the 1st phase. During the EA # 1 evolutionary stage, weights of the Grossberg layer are tuned, whereas the Kohonen layer is kept fixed (no crossover or mutation). For the recombination in the Grossberg layer we used 100% probability. As typically depicted in figure 4, we employed: (Wyd',Wya')=SBX(Wyd,Wya); (Wzd',Wza') = SBX(Wzd,Wza). The mutation in the Grossberg layer is done with polynomial mutation. Once the 1st phase is terminated, the 2nd one starts with a new counting of the generation number, and with a new termination criterion. The 2nd phase starts with uniting the groups into one population. In *Interact* # 2a updates are done only for the offspring. Following Interact # 2b, F1 and F2 are calculated for each individual based on the accumulated scoring. In contrast to the crossover procedure of [9], in *EA # 2* we employ a special mating procedure, where crossover is done by comparing classification neurons (Kohonen neuron weight vectors) in the mating.

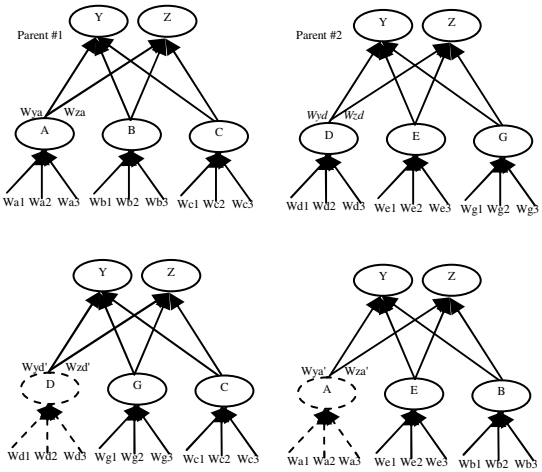


Fig. 4. Mating and mutating CPNs

parents to ensure that crossover is done among weights that are connected to Kohonen neurons that represent the "same" class. First, one neuron of one of the parents is selected, and crossover is performed with the closest neuron in the second parent. Next, another neuron of the first parent is selected and crossover is performed with the closest neuron that wasn't selected before. This procedure is continued for the rest of the neurons of the Kohonen layer. In the above, the term crossover between two neurons means that the Grosberg weights from the neurons are crossed-over using the SBX approach. The crossover operation is used here with a chance of 50%. No mutation used in the Kohonen layer. The mutation in the Grossberg layer is done with polynomial mutation. Figure 4 depicts an example of two parents (top) and their two offspring (bottom). Assume that among neurons D, E and G, neuron D has the closest weight vector (Kohonen layer) to the weight vector of neuron A. Similarly assumes G is the closest to B and E is the closest to C. Crossover occurred only for the first two cases due to the 50% chance for crossover. A "*Termination criterion # 2"* is used to terminate the second phase of the algorithm. In our study a maximal number of generations is used (currently 250).

4 Experimental Study

As discussed in [9], due to a local Pareto phenomenon, different runs are expected to produce different fronts. To compare the results we use the s-measure for the fronts, and plot the statistical results as boxplots from 30 different runs per each type of an algorithm. Figures 5, 6, 7, present the results for the old CPN of [9], for the current (new) CPN and for FFN, respectively. Comparing the medians it is concluded that both the old and the new CPN provide better results than FFN. Also, it is clear that the new version of CPN converges faster than the old one. Figures 8 & 9 show the best

F2 paths, which are obtained using FFN and CPN respectively. These paths are obtained using 400 allowable steps. The superiority of the CPN controller is clearly observed when comparing figures 8 &9. From figures 8 & 9, it becomes evident that the best F2 controller runs the robot into narrow places, on the expense of safety, while striving to reach all targets. Figure 10 shows a "typical best F1 path", as obtained by both CPN and FFN. When observing figure 10, recall that the concept behind the best F1 controller is to have a safe-straight moving robot. This means that the robot is expected to be attracted to spacey areas, where it can move straight and far from obstacles, rather than to the targets. In contrast to figures 8 & 9 the shown path of figure 10 clearly avoids narrow areas where most of the targets are. The best F1 controller "prefers" using the available steps on the larger, hence safer, yet empty room.

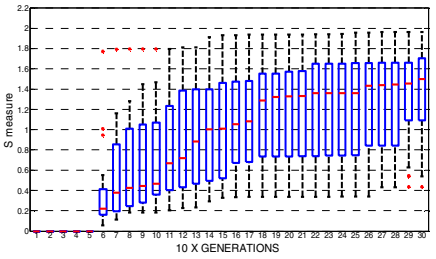


Fig. 5. Results of old-CPN

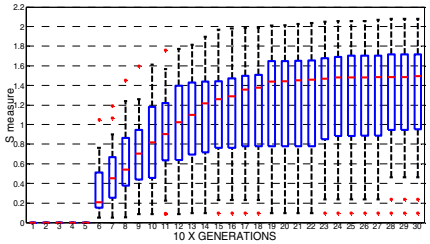


Fig. 6. Results of new-CPN

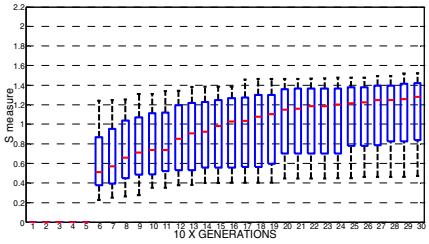


Fig. 7. Results of FFN

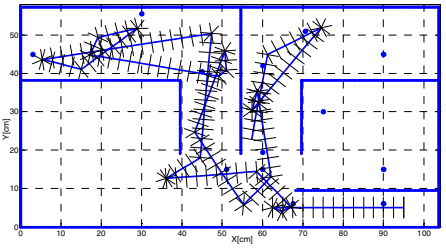


Fig. 8. Best F2 path of FFN

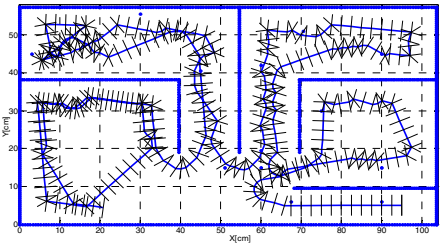


Fig. 9. Best F2 path of CPN

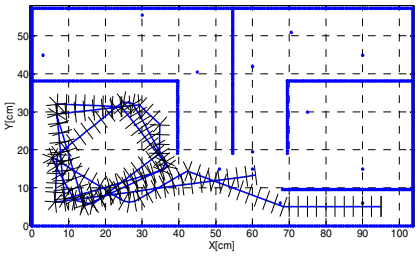


Fig. 10. Best F1 typical path CPN/FFN

5 Conclusions

The main conclusion of [9] is that CPNs cope well with evolving controllers for multi-objective navigation problem and with adaptation to new environment. While in [9] the primary focus is on the demonstration of the proposed algorithm, here, for the first time, it is compared with an FFN approach. Based on the medians of the results, both the CPN version of [9] and the current one appear superior to FFN. In addition, the current CPN version appears to have a faster convergence when compared with the older version. Due to the large variances, further runs and tests are needed to statistically substantiate these conclusions.

The study in [9] and its extension here appear to be the first to employ CPNs for ER applications. Much work is left on the methodology and on the computational aspects of evolving CPNs as compared with current approaches. Future studies should include issues such as: (a) more challenging problems, (b) studying the classification scheme with increasing complexity, and (c) actual implementation.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.: A Fast and Elitist Multi Objective Genetic Algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002)
2. Floreano, D., Mondada, F.: Evolution of Homing Navigation in a Real Mobile Robot. Systems, Man and Cybernetics, Part B 26(3), 396–407 (1996)
3. Grossberg, S.: Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control, SERBIULA, Venezuela (1982)
4. Han, S.J., Oh, S.Y.: An Optimized Modular Neural Network Controller Based on Environment Classification and Selective Sensor Usage for Mobile Robot Reactive Navigation. Neural Comput. Appl. 17(2), 161–173 (2008)
5. Hecht-Nielsen, R.: Counterpropagation Networks. Applied Optics 26(23), 4979–4984 (1987)
6. Israel, S., Moshaiov, A.: Bootstrapping Aggregate Fitness Selection with Evolutionary Multi-objective Optimization. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 52–61. Springer, Heidelberg (2012)
7. Kohonen, T.: Self-Organizing Feature Maps and Abstractions. In: 3rd Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots, pp. 39–45 (1984)
8. Moshaiov, A., Ashram-Wittenberg, A.: Multi-objective Evolution of Robot Neuro-Controllers. In: CEC 2009, Proceedings of the 11th Conference on Congress on Evolutionary Computation, pp. 1093–1100. IEEE Press, Piscataway (2009)
9. Moshaiov, A., Zadok, M.: Evolution of CPN Controllers for Multi-objective Robot Navigation in Various Environments. In: Proc. of the Int. Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems, ERLARS (2012)
10. Mouret, J.B., Doncieux, S.: Overcoming the Bootstrap Problem in Evolutionary Robotics using Behavioral Diversity. In: CEC 2009, Proceedings of the 11th Conference on Congress on Evolutionary Computation, pp. 1161–1168. IEEE Press, Piscataway (2009)
11. Yao, X.: Evolving Artificial Neural Networks. Proc. IEEE 87(9), 1423–1447 (1999)

Toward Automatic Gait Generation for Quadruped Robots Using Cartesian Genetic Programming

Kisung Seo[1] and Soohwan Hyun[2]

[1] Dept. of Electronic Engineering, Seokyeong University, Seoul, Korea
[2] Hyundai Heavy Industries Research Institute, Yongin, Korea

Abstract. This paper introduces a new gait generation method for quadruped robots using CGP (Cartesian Genetic Programming) based on refinement of regression polynomials for a joint trajectory. CGP uses as genotype a linear string of integers that are mapped to a directed graph. Therefore, some evolved modules for regression polynomials in CGP can be shared and reused among multiple outputs for joint trajectories. To investigate the effectiveness of the proposed approach, experiments on gaits were executed for a Bioloid quadruped robot in the Webots environment.

Keywords: Cartesian Genetic Programming, Gait Generation, Sharing Modules, Quadruped Robots Walking.

1 Introduction

The mobility of a walking robot is distinguished from that of a wheeled robot by its ability to traverse uneven and unstructured environments [14]. Quadruped robots, such as the Sony Aibo [5,6], an entertainment pet, and Bigdog [11], a practical field robot, are more stable than humanoid robots in locomotion. Providing good locomotion capabilities for robots is very significant for allowing them to carry out useful tasks in a variety of environments.

Automatic generation of gaits is especially important for walking robots because different environments and newly developed robots make it necessary to generate a variety of gaits in a short period of time. Gaits may be optimized for different properties, including fast velocity and/or high stability, and for specific requirements such as highest or lowest posture, and for various other traits. Planning gaits for quadruped robots is a challenging task that requires optimizing the locus of the robot's paw, an initial position, and a number of steps, in a highly irregular and multidimensional space. There is much research that imitates walking of animals such as dogs and horses [3].

In order to obtain the maximum speed of walking without falling, evolutionary approaches have been applied. A HyperNEAT approach [15] that generate gaits by evolving neural networks is introduced. Most of them are GA-based approaches [1,2,4,6] that seek to optimize a pre-selected set of parameters, such as locus of paw, initial position. A GP-based approach using a symbolic regression for generating joint trajoctory has also been introduced to improve on some constraints of GA-based

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 599–605, 2013.
© Springer-Verlag Berlin Heidelberg 2013

methods optimizing joint trajectories and shown better results [12,13]. Although symbolic regression approach using GP is quite useful to generate a gait in joint space, but it may have some lacks of efficiency for search structually fine tuning of building blocks or modules.

CGP [9,10] is a different form of Genetic Programming [7,8] in which a program is represented as a network of nodes. The nodes are combined to express a function. Previous columns of nodes may have their outputs connected to a node in the current column. That means some useful building blocks can be connected to nodes in multiple times. Therefore, it can search more efficiently on some particular types of problems which require certain relationships among outputs, such as partially similar trajectories of single or multiple joints. This concept is slightly different with ADF of GP. Whereas an ADF represents a complete module, a similar object in CGP is amorphous type. One of the attractive features of CGP is that no explicit encoding is required to facilitate this [9,10]. Thus it is possible to refine building blocks of natures of gait inherently through cascade connections from inputs to outputs.

In this paper, we propose a new CGP (Cartesian Genetic Programming)-based gait generation method and investigate a possibility of seeking meaningful modules for gait generations. To investigate the effectiveness of the approach, experiments are performed and analyzed for the Bioloid quadruped robot in the Webots environment.

2 Cartesian Genetic Programming

CGP [9,10] is a different form of Genetic Programming [7,8] in which a program is represented as an indexed graph. The graph is encoded in the form of a linear string of

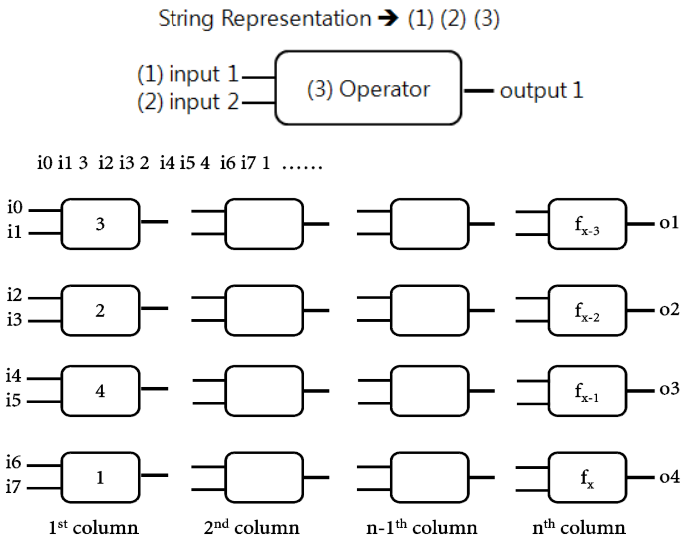


Fig. 1. Representation of Cartesian Genetic Programming

integers. Each block of a CGP can be represented as 3 strings (inputs 1, 2 and operator 3) as shown in Figure 1. The blocks are combined to express a function. Nodes in the same column are not allowed to be connected to each other, and any node may be either connected or disconnected. Unlike standard GP using tree expressions, CGP can obtain multiple outputs using a linear string of integers, and additionally can have features that constitute re-usable modules.

3 CGP Based Gait Control

This section explains how we develop a fast gait for a quadruped robot using Cartesian genetic programming (CGP) in joint spaces. In Figure 2, the value of the first input (i0) is X, which expresses time, and the other inputs are random constants. The output of the evaluation of a CGP genotype corresponds to trajectories of each joint. The cascade connections from inputs to outputs construct polynomial networks for representing a set of trajectories of gait.

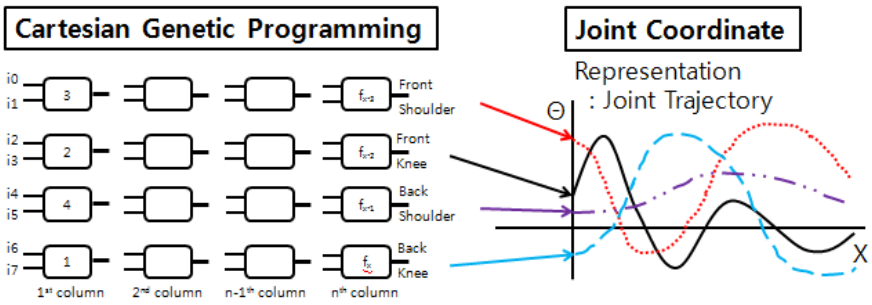


Fig. 2. CGP Based Gait Generation Method

Node outputs may be widely re-used one can consider it as employing Automatic Re-used Outputs (AROs) [9]. GP provide a similar concept of Automatically Defined Functions (ADFs), but they are slightly different from AROs of CGP. Whereas ADFs represent a complete module, AROs show an amorphous module. That means the module by ARO is more practical than ADF, because a complete shape of module is hard to be emerged in real situations. Most of useful building blocks have different forms in a little bit each other, even if they belong to the same kind. Especially, it is more natural that the regression polynomials are evolved in a cascaded connection of Re-used Outputs rather than using Defined Functions for the joint trajectories of robots.

Another difference is that an ADF should be evolved simultaneously with main tree in GP, thus search efforts should be distributed to both parts. That requires much computational costs. However an ARO in CGP is that no explicit encoding is required to facilitate this [9]. That will be one of the attractive features of CGP based approach for generation gait of robots considering quite large amount of simulation time.

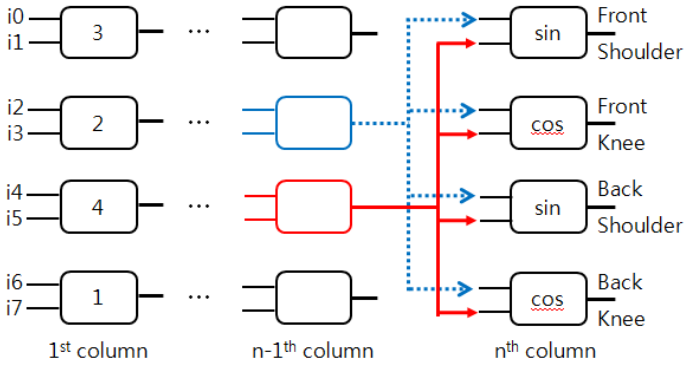


Fig. 3. Sharing of modules in CGP

A representative illustration of expected advantages of CGP is sharing a trajectory graph with different phases in gait generation. As shown in Figure 3, the second and third row modules in the n-1th column can be used as common inputs of an arithmetic function in the last column that generates joint trajectories for each leg. Therefore, the CGP-based gait generation method can search more efficiently some useful modules and relationships among outputs being evolved for different legs.

4 Experiments and Analysis

This section describes how we evaluated the proposed approach to develop a fast gait for a quadruped robot using CGP. A simulated model of Bioloid in Webots was used, as shown in Figure 4.

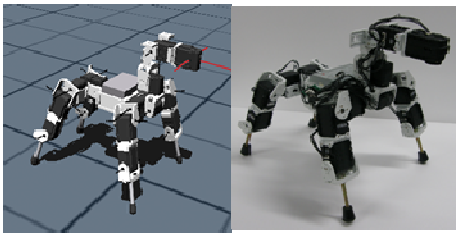


Fig. 4. Simulation model and real Bioloid quadruped robot

The Webots [5] mobile robotics simulation software developed by Cyberbotics provides the user with a rapid prototyping environment for modeling, programming and simulating mobile robots. Webots relies on ODE (Open Dynamics Engine) to perform accurate dynamic physics simulation. The fitness function of gait generation is defined to obtain the joint trajectory set that provides the fastest walking with only

a small sideways diversion. A trot gait is selected, as used in other approaches. The CGP parameters were as shown below in Table 1.

Table 1. The CGP parameters

CGP Parameters
Terminal set : Random Constants, X Function set : SIN, COS, +, -, *, / Number of generations : 100 Mutation : 0.5 Number of rows : 4 Number of columns : 200 Levels back : 200 4 number of rows 200 number of columns 200 levels back

The tabular results of velocities for generated gaits are provided in Table 2. Every run was repeated 10 times for each case. The max velocity value from CGP was 27.03 cm/s, obtained with population size 500 and it is very competitive. The results showed increased max velocities as population size was increased. In this paper, we have not attempted to compare the performances to the results of other approaches, rather we have demonstrated the effectiveness of our methodology.

Table 2. Experimental results of velocities for various population sizes

	Popsize	**Average Velocity(cm/s)**	**Max Velocity(cm/s)**
Cartesian GP	150	12.67	19.68
	250	13.24	23.70
	500	17.35	27.03

The motions of obtained gaits for the best evolved solutions of CGP approach are displayed in Figure 5.

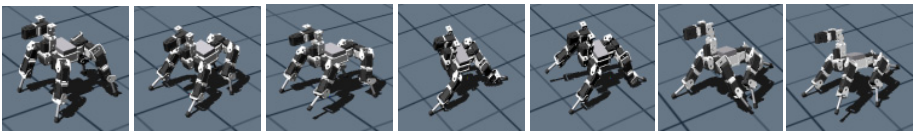


Fig. 5. The best gait movements of CGP method

5 Conclusions

We have developed a new and efficient automatic gait generation method based on CGP (Cartesian Genetic Programming). The main idea of the method is that CGP can generate multiple outputs through one genotype expression and can share some amorphous modules in generation of the trajectory of more than one joint using Automatic Re-used Outputs (AROs). Therefore, it can search more efficiently on some particular types of problems which require certain relationships among outputs, such as partially similar trajectories of single or multiple joints. It is also possible to refine building blocks of natures of gait through cascade connections from inputs to outputs.

To demonstrate the effectiveness of our proposed approach, experiments on the automatic gait generation of a quadruped robot were performed. The experimental results showed the possibilities for re-using and/or sharing of modules in gait evolution.

Acknowledgements. This work was supported by National Research Foundation of Korea Grant funded by the Korea government (NRF-2011-0009958).

References

1. Akın, H.L., Meriçli, Ç., Meriçli, T., Kaplan, K., Çelik, B.: Cerberus'05 Team Report, Technical Report, Boğaziçi University (2005)
2. Chen, W.: Odometry Calibration and Gait Optimisation. Technical Report. The University of New South Wales, School of Computer Science and Engineering (2005)
3. Doan, P., Vo, H., Kim, H., Kim, S.: A New Approach for Development of Quadruped Robot Based on Biological Concepts. International Journal of Precision Engineering and Manufacturing 11, 559–568 (2010)
4. Dong, H., Zhao, M., Zhang, J., Shi, Z., Zhang, N.: Gait planning of quadruped robot based on third-order spline interpolation. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2006), pp. 5756–5761. IEEE Press, China (2006)
5. Hohl, L., Tellez, R., Michel, O., Ijspeert, A.J.: Aibo and Webots: Simulation, wireless remote control and controller transfer. Robotics and Autonomous Systems 54, 472–485 (2006)
6. Hornby, G.S., Takamura, S., Yamamoto, T., Fujita, M.: Autonomous evolution of dynamic gaits with two quadruped robots. IEEE Trans. Robotics 21, 402–410 (2005)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge (1992)
8. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge (1994)
9. Miller, J.F., Thomson, P.: Cartesian Genetic Programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
10. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the Evolutionary Design of Digital Circuits - Part I. Genetic Programming and Evolvable Machines 1, 8–35 (2000)
11. Raibert, M., Blankespoor, K., Nelson, G., Playter, R.: Bigdog, the rough-terrain quadruped robot. Technical report, Boston Dynamics (2008)

12. Seo, K., Hyun, S.: A Comparative Study between Genetic Algorithm and Genetic Programming Based Gait Generation Methods for Quadruped Robots. 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, Part I. LNCS, vol. 6024, pp. 352–360. Springer, Heidelberg (2010)
13. Seo, K., Hyun, S., Goodman, E.: Genetic Programming-Based Automatic Gait Generation in Joint Space for a Quadruped Robot. Advanced Robotics 24, 2199–2214 (2010)
14. Tenreiro, J.A., Silva, M.F.: An Overview of Legged Robots. In: Proceedings of the International Symposium on Mathematical Methods in Engineering, pp. 27–29 (2006)
15. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, C.J., Lipson, H.: Evolving Robot Gaits in Hardware: the HyperNEAT Generative Encoding Vs. Parameter Optimization, Advances in Artificial Life. In: Proceedings of the 11th European Conference on the Synthesis and Simulation of Living Systems, pp. 890–897 (2011)

Adapting the Pheromone Evaporation Rate in Dynamic Routing Problems

Michalis Mavrovouniotis and Shengxiang Yang

School of Computer Science and Informatics, De Montfort University
The Gateway, Leicester LE1 9BH, United Kingdom
m.mavrovouniotis@hotmail.com, syang@dmu.ac.uk

Abstract. Ant colony optimization (ACO) algorithms have proved to be able to adapt to dynamic optimization problems (DOPs) when stagnation behaviour is avoided. Several approaches have been integrated with ACO to improve its performance for DOPs. The adaptation capabilities of ACO rely on the pheromone evaporation mechanism, where the rate is usually fixed. Pheromone evaporation may eliminate pheromone trails that represent bad solutions from previous environments. In this paper, an adaptive scheme is proposed to vary the evaporation rate in different periods of the optimization process. The experimental results show that ACO with an adaptive pheromone evaporation rate achieves promising results, when compared with an ACO with a fixed pheromone evaporation rate, for different DOPs.

1 Introduction

Ant colony optimization (ACO) algorithms have shown good performance when applied to difficult optimization problems under static environments [2]. However, in many real-world applications, we have to deal with dynamic environments, where the optimum changes and needs re-optimization. It is believed that ACO algorithms can adapt to dynamic changes since they are inspired from nature, which is a continuous adaptation process [7]. In practice, they can adapt by transferring knowledge from past environments, using the pheromone trails, to speed up re-optimization. The challenge to such algorithms lies in how quickly they can react to dynamic changes in order to maintain the high quality of output instead of early stagnation behaviour, where all ants construct the same solutions and lose their adaptation capabilities.

Developing strategies for ACO algorithms to deal with stagnation behaviour and address dynamic optimization problems (DOPs) has attracted a lot of attention, which includes local and global restart strategies [6], memory-based approaches [5], pheromone manipulation schemes to maintain diversity [3], and immigrants schemes to increase diversity [8].

The adaptation capabilities of ACO rely on the pheromone evaporation where a constant amount of pheromone is deducted to eliminate pheromone trails that represent bad solutions that may bias ants to search to the non-promising areas of the search space. In this paper, the impact of the pheromone evaporation rate

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 606–615, 2013.
© Springer-Verlag Berlin Heidelberg 2013

is examined on DOPs, and an adaptive scheme is designed for ACO. Adaptive methods have been successfully applied for different parameters of ACO, including the evaporation rate [10,12]. However, these methods have been investigated on static optimization problems.

The rest of the paper is organized as follows. Section 2 describes the generation of dynamic routing DOPs. Section 3 describes an ACO algorithm and gives details for its adaptation capabilities in DOPs. Section 4 describes the proposed scheme where the evaporation rate in ACO is adapted. Section 5 describes the experiments carried out on a series of different DOPs. Finally, Section 6 concludes this paper with directions for future work.

2 Generating Dynamic Routing Environments

Routing problems are usually illustrated using weighted graphs. Let $G = (V, E)$ be a weighted graph where V is a set of n nodes and E is a set of links. Each node i has a location defined by (x, y) and each link (i, j) is associated with a non-negative distance d_{ij}. Usually, the distance matrix of a problem instance is defined as $\boldsymbol{D} = (d_{ij})_{n \times n}$.

In order to generate dynamic routing problems, the dynamic benchmark generator for permutation-encoded problems (DBGP) [9] is used, which converts any static problem instance to a dynamic environment. In case the optimum of the static problem instance is known, then it will remain known during the environmental changes, because DBGP biases algorithms to search to a new location in the fitness landscape, instead of modifying the fitness landscape.

Every f iterations a random vector $\boldsymbol{r}(T)$ is generated that contains all the objects of a problem instance of size n, where $T = \lceil t/f \rceil$ is the index of the period of change, t is the iteration count of the algorithm, and f determines the frequency of change. The magnitude m of change depends on the number of swapped locations of objects. More precisely, $m \in [0.0, 1.0]$ defines the degree of change, in which only the first $m \times n$ of $\boldsymbol{r}(T)$ object locations are swapped. Then a randomly re-ordered vector $\boldsymbol{r'}(T)$ is generated that contains only the first $m \times n$ objects of $\boldsymbol{r}(T)$. Therefore, exactly $m \times n$ pairwise swaps are performed using the two random vectors.

3 ACO in Dynamic Environments

3.1 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System

The ACO metaheuristic consists of a population of μ ants where they construct solutions and share their information among each other via their pheromone trails. The first ACO algorithm developed is the Ant System (AS) [1]. Many variations of the AS have been applied to difficult optimization problems [2].

One of the best performing ACO algorithm is the $\mathcal{MAX}$-$\mathcal{MIN}$ AS ($\mathcal{MM}$AS) [11]. Ants read and write pheromones in order to construct their solutions.

Each ant k uses a probabilistic rule to choose the next city to visit. The decision rule an ant k uses to move from city i to city j is defined as follows:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in \mathcal{N}_i^k, \tag{1}$$

where τ_{ij} is the existing pheromone trail between cities i and j, $\eta_{ij} = 1/d_{ij}$ is the heuristic information available a priori, where d_{ij} is the distance between cities i and j. $\mathcal{N}_i^k$ denotes the neighbourhood of cities for ant k when the current city is i. α and β are the two parameters which determine the relative influence of τ and η, respectively. The pheromone trails in $\mathcal{MM}$AS are updated by applying evaporation as follows:

$$\tau_{ij} \leftarrow (1 - \rho)\, \tau_{ij}, \forall (i, j), \tag{2}$$

where ρ is the evaporation rate which satisfies $0 < \rho \leq 1$, and τ_{ij} is the existing pheromone value. After evaporation the best ant deposits pheromone as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \forall (i, j) \in T^{best}, \tag{3}$$

where $\Delta\tau_{ij}^{best} = 1/C^{best}$ is the amount of pheromone that the best ant deposits and C^{best} defines the solution quality of tour T^{best}. Since only the best ant deposits pheromone, the algorithm will quickly converge towards the best solution of the first iteration. Therefore, pheromone trail limits are imposed in order to avoid this behaviour.

3.2 Response to Dynamic Changes

ACO algorithms are able to use knowledge from previous environments using the pheromone trails generated in the previous iterations. For example, when the changing environments are similar, the pheromone trails of the previous environment may provide knowledge to speed up the optimization process to the new environment. However, the algorithm needs to be flexible enough to accept the knowledge transferred from the pheromone trails, or eliminate the pheromone trails, in order to adapt well to the new environment.

ACO algorithms can be applied directly to DOPs without any modifications due to the pheromone evaporation. Lowering the pheromone values enables the algorithm to forget bad decisions made in previous iterations. When a dynamic change occurs, evaporation eliminates the pheromone trails of the previous environment from areas that are not visited frequently and may bias ants not to adapt well to the new environment.

The adaptation via pheromone evaporation may be a sufficient choice when the changing environments are similar, otherwise a complete re-initialization of the pheromone trails after a dynamic change occurs may be a better choice. However, such action is available only in DOPs where the frequency of change is available beforehand or in DOPs where the dynamic changes can be detected. In our case, the dynamic changes can be detect by re-evaluating some stored solutions, used as detectors, in every iteration [8].

4 ACO with Adaptive Evaporation Rate

4.1 Effect of the Pheromone Evaporation Rate

Although ACO has adaptation capabilities due to the pheromone evaporation, the time required to adapt to the new environment may depend on the problem size and the magnitude of change. When the environmental change is severe then it may take longer to eliminate unused pheromone trails, therefore a high evaporation rate may be more suitable. More precisely, a high evaporation rate will eliminate the high intensity of pheromone trails that are usually concentrated to the optimum of the previous environment that is caused by stagnation behaviour. On the other hand, a high pheromone evaporation rate may destroy information that can be used on further environments, since any bad solution in the current environment may be good in the next environment.

In traditional ACO algorithms the evaporation rate is usually fixed. A low evaporation rate corresponds to slow adaptation, whereas a high evaporation rate corresponds to fast adaptation. However, we believe that a fixed evaporation rate is not the best choice when addressing DOPs, since at different stages of the optimization process for different optimization problems and under different dynamic environments, the most appropriate evaporation rate varies.

4.2 Detect Stagnation Behaviour

In order to adapt the value of the pheromone evaporation during the search process, the exploration of the algorithm is measured in order to detect stagnation behaviour. A direct way that can give an indication of exploration is to measure the diversity of the solutions. The measurement for routing problems is usually based on the common edges between the solutions [8]. Such a measure may be computational expensive since there are $\mathcal{O}(n^2)$ possible pairs to be compared and each single comparison has a complexity of $\mathcal{O}(n)$.

A more efficient measurement is the λ-branching factor [4], which measures the distribution of the pheromone trail values. The idea of λ-branching is described as follows: If for a given object $i \in V$, the concentration of pheromone trails on almost all the incident arcs becomes very small but is large for a few others, then the freedom of exploring other paths from object i is very limited. Therefore, if this situation arises simultaneously for all objects of graph G, the search space that is searched by ants becomes relatively small.

The average $\bar{\lambda}(t)$ branching factor at iteration t is defined as follows:

$$\bar{\lambda}(t) = \frac{1}{2n} \sum_{i=1}^{n} \lambda^i, \tag{4}$$

where n is the number of objects in the corresponding graph and λ^i is the λ-branching factor for object i, which is defined as follows:

$$\lambda^i = \sum_{j=1}^{d} L_{ij} \tag{5}$$

where d is the number of available arcs incident to object i and L_{ij} is defined as follows:

$$L_{ij} = \begin{cases} 1, & \text{if } (\tau^i_{min} + \lambda(\tau^i_{max} - \tau^i_{min})) \leq \tau_{ij}, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

where λ is a constant parameter ($\lambda = 0.05$ by default [4]), τ^i_{min} and τ^i_{max} are the minimum and maximum pheromone trail values on the arcs incident to object i, respectively. A value of $\bar{\lambda}(t)$ close to 1 indicates stagnation behaviour.

4.3 Adapting Pheromone Evaporation Rate

Considering the statements above, if the algorithm reaches stagnation behaviour, the evaporation rate needs to be increased in order to eliminate the high intensity of pheromone trails in some areas and increase exploration. However, very high exploration may disturb the optimization process because of randomization [8].

According to the behaviour of the algorithm in terms of searching, we have the following pheromone evaporation rate update rule:

$$\rho(t) = \begin{cases} \rho(t-1) - \sigma, & \text{if } \bar{\lambda}(t) > 1, \\ \rho(t-1) + \sigma, & \text{otherwise.} \end{cases} \tag{7}$$

where $\bar{\lambda}(t)$ is defined in Eq. (4) and σ is the step size of varying the evaporation rate ρ at iteration t. A good value of σ was found to be 0.001 because a higher value may quickly increase ρ to an extreme evaporation rate and destroy information and a smaller value may not have any effect to the performance of ACO.

5 Experimental Study

5.1 Experimental Setup

In the experiments, we compare a $\mathcal{MM}$AS with a global re-initialization of the pheromone trails, denoted as $\mathcal{MM}\text{AS}_R$, and a $\mathcal{MM}$AS with the best fixed evaporation rate, denoted as $\mathcal{MM}\text{AS}_B$ against the $\mathcal{MM}$AS with the proposed adaptive pheromone evaporation, denoted as $\mathcal{MM}\text{AS}_A$. For all algorithms, we set $\alpha = 1$, $\beta = 5$, $q_0 = 0.0$, and $\mu = 50$, except for $\mathcal{MM}\text{AS}_R$ where $\mu = 50 - d_T$, where $d_T = 6$ is the number of detectors. The evaporation rate for $\mathcal{MM}\text{AS}_R$ was set to $\rho = 0.4$. For $\mathcal{MM}\text{AS}_B$ the best value from $\rho \in \{0.02, 0.2, 0.4, 0.6, 0.8\}$ was selected, whereas for $\mathcal{MM}\text{AS}_A$ ρ was adapted by Eq. (7).

For each algorithm on a DOP, $N = 30$ independent runs were executed on the same environmental changes. The algorithms were executed for $G = 1000$ iterations and the overall offline performance is calculated as follows:

$$\bar{P}_{offline} = \frac{1}{G} \sum_{i=1}^{G} \left(\frac{1}{N} \sum_{j=1}^{N} P^*_{ij} \right) \tag{8}$$

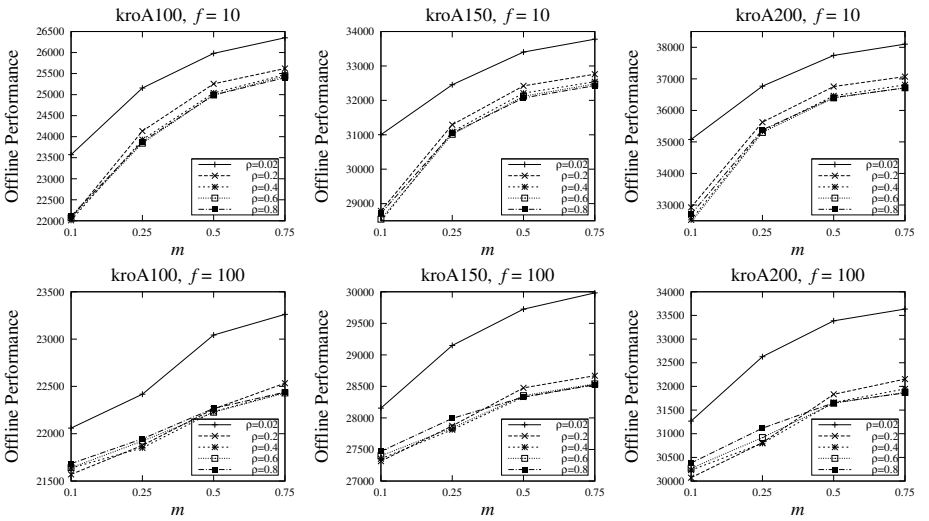


Fig. 1. Impact of the evaporation rate on the offline performance of a conventional $\mathcal{MM}$AS on different DOPs

where P_{ij}^* defines the tour cost of the best ant since the last dynamic change of iteration i of run j [7].

We took three travelling salesman problem (TSP) instances[1] and three vehicle routing problem (VRP) instnaces[2] as the base and used the DBGP described in Section 2 to generate DOPs. The value of f was set to 10 and 100, which indicate fast and slowly changing environments, respectively. The value of m was set to 0.1, 0.25, 0.5, and 0.75, which indicate the degree of environmental changes from small, to medium, to large, respectively. As a result, eight dynamic environments, i.e., 2 values of $f \times 4$ values of m, for each problem instance are generated to systematically analyze the adaptation and searching capability of algorithms on the DOPs.

5.2 Experimental Results and Analysis

The experimental results regarding the different ρ values for $\mathcal{MM}\text{AS}_B$ on dynamic TSPs are presented in Fig. 1. Note that the corresponding experimental results for dynamic VRPs show similar observations and are not presented here. The offline performance of the different algorithms on dynamic TSPs and dynamic VRPs and the corresponding statistical results of Wilcoxon rank-sum test, at the 0.05 level of significance are presented in Table 1 and Table 2, respectively. Moreover, the dynamic behaviour of the algorithms is presented in Fig. 2. From the experimental results, several observations can be made by comparing the behaviour of the algorithms.

[1] `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`.
[2] `http://neo.lcc.uma.es/vrp/`.

Table 1. Experimental results of the algorithms regarding the offline performance

Travelling Salesman Problem Instances								
	$f = 10$				$f = 100$			
$m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
Alg. & Inst.	`kroA100`(Optimum=21282)							
$\mathcal{MM}$AS$_B$	22010	23844	24989	25401	21570	21850	22227	22430
$\mathcal{MM}$AS$_A$	22069	23542	24448	24800	21683	21819	22049	22468
$\mathcal{MM}$AS$_R$	24576	24580	24583	24588	22244	22252	22224	22212
Alg. & Inst.	`kroA150`(Optimum=26524)							
$\mathcal{MM}$AS$_B$	28488	31013	32070	32426	27315	27814	28330	28526
$\mathcal{MM}$AS$_A$	28690	30507	31444	31778	27299	27726	28140	28262
$\mathcal{MM}$AS$_R$	31520	31526	31515	31516	28208	28204	28215	28198
Alg. & Inst.	`kroA200`(Optimum=29368)							
$\mathcal{MM}$AS$_B$	32454	35300	36394	36711	30071	30796	31644	31863
$\mathcal{MM}$AS$_A$	32353	34524	35560	35872	30167	30645	31245	31506
$\mathcal{MM}$AS$_R$	35375	35368	35362	35375	31282	31368	31338	31380
Vehicle Routing Problem Instances								
	$f = 10$				$f = 100$			
$m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
Alg. & Inst.	`F-n45-k4`(Optimum=724)							
$\mathcal{MM}$AS$_B$	800.42	807.56	816.98	823.20	792.79	796.24	796.33	797.65
$\mathcal{MM}$AS$_A$	800.55	805.22	814.02	820.39	795.63	797.66	797.22	799.04
$\mathcal{MM}$AS$_R$	812.16	812.06	812.34	812.16	799.76	800.46	799.55	799.70
Alg. & Inst.	`F-n72-k4`(Optimum=237)							
$\mathcal{MM}$AS$_B$	268.69	279.67	285.76	288.36	259.95	263.00	265.82	266.81
$\mathcal{MM}$AS$_A$	270.36	281.13	286.47	289.20	261.17	263.76	267.18	267.44
$\mathcal{MM}$AS$_R$	291.09	291.14	291.14	291.23	270.64	270.91	271.11	270.61
Alg. & Inst.	`F-n135-k7`(Optimum=1162)							
$\mathcal{MM}$AS$_B$	1298.71	1339.50	1365.33	1375.41	1255.00	1271.66	1286.62	1291.91
$\mathcal{MM}$AS$_A$	1297.46	1335.77	1363.29	1372.89	1255.21	1269.49	1283.98	1288.14
$\mathcal{MM}$AS$_R$	1348.90	1348.65	1348.87	1348.85	1281.08	1283.68	1283.35	1282.20

First, when the evaporation rate is set to $\rho = 0.02$, which is the recommended value for $\mathcal{MM}$AS on static problems [2, p. 71], has the worst results on DOPs, as observed from Fig. 1. Furthermore, a high evaporation rate, i.e., $\rho \geq 0.4$, often achieves better performance when $f = 10$. This is natural because when the environment changes quickly, a fast adaptation is required. When $f = 100$, the evaporation rate depends on the magnitude of change, e.g., when $m = 0.1$, $\rho = 0.2$ shows better performance. However, as the magnitude of change increases, a higher value of ρ achieves better performance. This validates our claim that the time required for ACO, in which pheromone evaporation is used, to adapt to the new environment depends on the magnitude of change.

Second, $\mathcal{MM}$AS$_B$ is outperformed by $\mathcal{MM}$AS$_R$ on most DOPs with $m = 0.5$ and $m = 0.75$, whereas the former outperforms the latter on all DOPs with $m = 0.1$ and $m = 0.25$; see the comparisons $\mathcal{MM}$AS$_B \Leftrightarrow \mathcal{MM}AS_R$ in Table 2.

Table 2. Statistical tests of comparing algorithms regarding the offline performance, where "+" or "−" means that the first algorithm is significantly better or the second algorithm is significantly better, respectively, and "∼" means that the algorithms are not significantly different

Travelling Salesman Problem Instances												
Alg. & Inst.	kroA100				kroA150				kroA200			
$f = 10, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_B$	−	+	+	+	−	+	+	+	+	+	+	+
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	+	−	+	+	+	−	+	+	−	−
$\mathcal{MM}\text{AS}_B \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	−	−	+	+	−	−	+	+	−	−
$f = 100, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_B$	−	∼	+	∼	∼	+	+	+	∼	+	+	+
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	+	−	+	+	+	∼	+	+	+	−
$\mathcal{MM}\text{AS}_B \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	∼	−	+	+	−	−	+	+	−	−
Vehicle Routing Problem Instances												
Alg. & Inst.	F-n45-k4				F-n72-k4				F-n135-k7			
$f = 10, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_B$	∼	+	+	+	−	−	−	−	∼	+	+	+
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	−	−	+	+	+	+	+	+	−	−
$\mathcal{MM}\text{AS}_B \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	−	−	+	+	+	+	+	+	−	−
$f = 100, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_B$	∼	∼	∼	−	−	∼	−	∼	∼	∼	∼	∼
$\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	+	∼	+	+	+	+	+	+	∼	−
$\mathcal{MM}\text{AS}_B \Leftrightarrow \mathcal{MM}\text{AS}_R$	+	+	+	−	+	+	+	+	+	+	∼	−

This is because when the environments are similar, due to a slight change, the pheromone trails of the previous environment help to start the optimization process from a promising area in the search space, whereas when the environments are different, due to a severe change, the pheromone trails of the previous environment mislead the searching to non-promising areas. This validates our claim that the adaptation of pheromone evaporation is useful when the environments are similar and useful knowledge can be transferred.

Finally, the proposed $\mathcal{MM}\text{AS}_A$ outperforms $\mathcal{MM}\text{AS}_B$ on most TSP DOPs when $m = 0.25$, $m = 0.5$ and $m = 0.75$, whereas the former is comparable with the latter when $m = 0.1$; see the comparisons $\mathcal{MM}\text{AS}_A \Leftrightarrow \mathcal{MM}\text{AS}_B$ in Table 2. This is probably because when the evaporation is high, it may destroy useful knowledge from the previous environment after a dynamic change. Therefore, a low evaporation rate sometimes may be a better choice, even when the dynamic change is severe, for the first iterations after a dynamic change to obtain knowledge, and a higher evaporation rate may be a better choice later on to avoid the stagnation behaviour. This can be observed from Fig. 2 where $\mathcal{MM}\text{AS}_A$ converges faster and to a better optimum than $\mathcal{MM}\text{AS}_B$. Fortunately, even if $\mathcal{MM}\text{AS}_A$ does not achieve the best result compared to $\mathcal{MM}\text{AS}_B$, e.g., when $f = 100$ for VRP DOPs, its performance level is still satisfactory since they are usually not significantly different. This can be expected since the

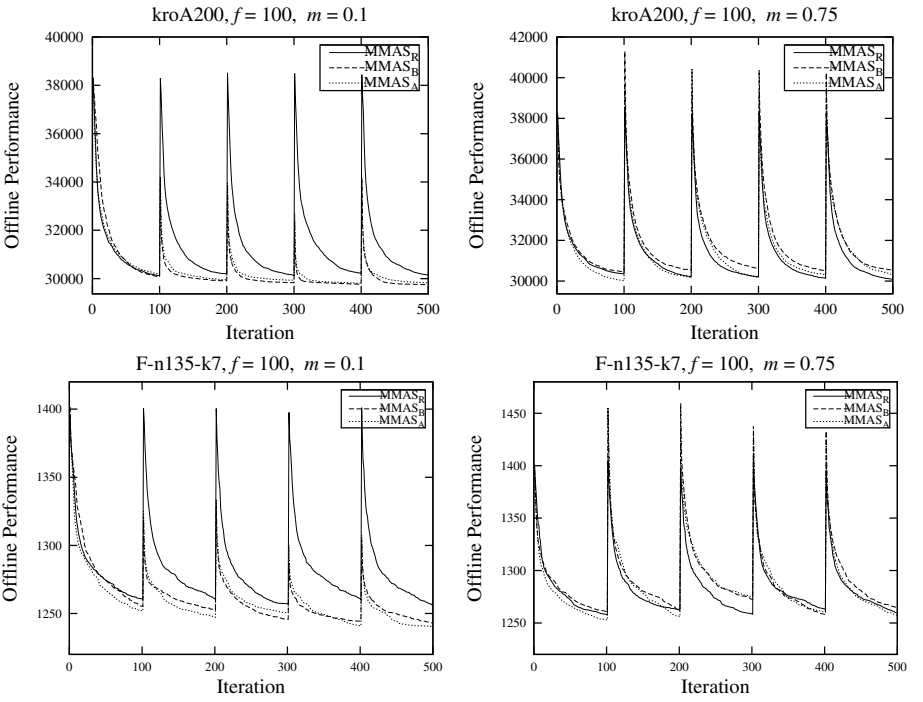


Fig. 2. Dynamic behaviour of the algorithms with respect to offline performance against the iterations in slowly changing environments for the first 500 iterations in different DOPs: 1) `kroA200` for TSP; and 2) `F-n135-k7` for VRP

results of $\mathcal{MM}\text{AS}_B$ are obtained via fine-tuning the evaporation rate. Moreover, $\mathcal{MM}\text{AS}_A$ outperforms $\mathcal{MM}\text{AS}_R$ in most DOPs, except when $m = 0.75$, because of the same reasons discussed for $\mathcal{MM}\text{AS}_B$ previously.

6 Conclusions

This paper examines the impact of the pheromone evaporation on the performance of ACO algorithms for DOPs. An adaptive evaporation rate is proposed for ACO to deal with DOPs, which is based on the detection of the stagnation behaviour. Experimental studies were performed on a series of DOPs to investigate the performance of the proposed approach. From the experimental results, several conclusions can be drawn. First, pheromone evaporation is important for ACO to address DOPs. Second, the higher the magnitude of the dynamic change, the higher the evaporation rate is needed. Third, the adaptation capabilities of pheromone evaporation perform well only when the environments are similar; otherwise, a re-initialization of the pheromone trails is required. Forth, the proposed adaptive evaporation rate promotes the performance of ACO in many routing DOPs but depends on the dynamics and the type of the DOP.

Finally, compared to the tedious work of fine-tuning the pheromone evaporation rate manually, the proposed adaptive scheme is more convenient and has sufficiently good performance under different conditions. However, the performance is slightly decreased in some cases, for the sake of this convenience.

For future work, it will be interesting to consider other ways for adapting the evaporation rate. Moreover, there are evidence that the more parameters adapted in ACO, the better the performance in optimization problems with static environment [10]. Therefore, another future work is to adapt more ACO parameters, e.g., α and β, in parallel for DOPs.

Acknowledgement. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1.

References

1. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. on Syst., Man and Cybern., Part B: Cybern. 26(1), 29–41 (1996)
2. Dorigo, M., Stützle, T.: Ant colony optimization. The MIT Press, London (2004)
3. Eyckelhof, C.J., Snoek, M.: Ant Systems for a Dynamic TSP: Ants Caught in a Traffic Jam. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) ANTS 2002. LNCS, vol. 2463, pp. 88–99. Springer, Heidelberg (2002)
4. Gambardella, M.L., Dorigo, M.: Ant-Q: A reinforcement learning approach to the traveling salesman problem. In: Proc of the 12th Int. Conf. on Machine Learning, pp. 252–260. Morgan Kaufmann (1995)
5. Guntsch, M., Middendorf, M.: Applying Population Based ACO to Dynamic Optimization Problems. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) ANTS 2002. LNCS, vol. 2463, pp. 111–122. Springer, Heidelberg (2002)
6. Guntsch, M., Middendorf, M.: Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoWorkshop 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
7. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. on Evol. 9(3), 303–317 (2005)
8. Mavrovouniotis, M., Yang, S.: Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem. In: Proc. of the 2012 IEEE Congress on Evol. Comput., pp. 2645–2652. IEEE Press (2012)
9. Mavrovouniotis, M., Yang, S., Yao, X.: A Benchmark Generator for Dynamic Permutation-Encoded Problems. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 508–517. Springer, Heidelberg (2012)
10. Pellegrini, P., Stützle, T., Birattari, M.: A critical analysis of parameter adaptation in ant colony optimization. Swarm Intelli. 6(1), 23–48 (2012)
11. Stützle, T., Hoos, H.: The $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System and local search for the traveling salesman problem. In: Proc. of the 1997 IEEE Int. Conf. on Evol. Comput., pp. 309–314. IEEE Press (1997)
12. Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., de Oca, M.M., Birattari, M., Dorigo, M.: Parameter adaptation in ant colony optimization, pp. 191–215. Springer, Heidelberg (2012)

Finding Robust Solutions to Dynamic Optimization Problems

Haobo Fu[1], Bernhard Sendhoff[2], Ke Tang[3], and Xin Yao[1]

[1] CERCIA, School of Computer Science, University of Birmingham, UK
[2] Honda Research Institute Europe, Offenbach, DE
[3] Joint USTC-Birmingham Research Institute in Intelligent Computation and Its Applications, School of Computer Science and Technology, University of Science and Technology of China, CN

Abstract. Most research in evolutionary dynamic optimization is based on the assumption that the primary goal in solving Dynamic Optimization Problems (DOPs) is Tracking Moving Optimum (TMO). Yet, TMO is impractical in cases where keeping changing solutions in use is impossible. To solve DOPs more practically, a new formulation of DOPs was proposed recently, which is referred to as Robust Optimization Over Time (ROOT). In ROOT, the aim is to find solutions whose fitnesses are robust to future environmental changes. In this paper, we point out the inappropriateness of existing robustness definitions used in ROOT, and therefore propose two improved versions, namely *survival time* and *average fitness*. Two corresponding metrics are also developed, based on which *survival time* and *average fitness* are optimized respectively using population-based algorithms. Experimental results on benchmark problems demonstrate the advantages of our metrics over existing ones on robustness definitions *survival time* and *average fitness*.

Keywords: Evolutionary Dynamic Optimization, Robust Optimization Over Time, Population-Based Search Algorithms.

1 Introduction

Applying population-based search algorithms to solving Dynamic Optimization Problems (DOPs) has become very active [6,13] as most real-world optimization problems are subject to environmental changes. DOPs deal with optimization problems whose specifications change over time, and the algorithm for DOPs needs to react to those changes during the optimization process as time goes by [9]. So far, most research on DOPs falls into the category of Tracking Moving Optimum (TMO) [2,8,11,12]. Recently, a more practical way of formulating DOPs, namely Robust Optimization Over Time (ROOT), has been proposed [5,7,14].

A DOP is usually represented as a dynamic fitness function $F(\overrightarrow{X}, \alpha(t))$, where $\overrightarrow{X}$ stands for the design variable and $\alpha(t)$ is the time-dependent problem parameters. $\alpha(t)$ can change continuously or discretely, and is often considered to be deterministic at any time point. In this paper, we investigate the case where

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 616–625, 2013.
© Springer-Verlag Berlin Heidelberg 2013

$\alpha(t)$ changes discretely. Hereafter, we use $F_t(\overrightarrow{X})$ to represent $F(\overrightarrow{X}, \alpha(t))$ for short. Briefly speaking, the objective in TMO is to optimize the current fitness function, while in ROOT solution's current and future fitnesses are both taken into consideration. To be more specific, if the current fitness function is $F_t(\overrightarrow{X})$, TMO is trying to find a solution maximizing [1] F_t, while ROOT aims at the solution whose fitness is not only good for F_t but also stays robust against future environmental changes.

A set of robustness definitions for solutions (A solution is the setting of design variable $\overrightarrow{X}$.) in ROOT have been proposed in [14] and used in [5,7]. Basically, those definitions consider solution's fitnesses over a time period. However, those definitions suffer from the following problems:

- All these robustness definitions are dependent on a fitness threshold parameter v, the setting of which requires the information of optimal solution in terms of current fitness at any time point. This limits the practical use of those robustness definitions, as most often the optimal solution for any time point is not known in real-world DOPs.
- A solution is considered 'robust' only if its fitness stays above v after an environmental change without any constraint on solution's current fitness. This might be inappropriate as robust solutions can have very bad fitnesses for current fitness function. This inappropriateness is reflected in the poor average fitness of robust solution in the experimental results in [7].
- Robustness definitions based on v only measure one aspect of robust solutions for DOPs. For example, solutions which have good average fitness over a certain time window could also be considered robust without any constraint on the fitness at any time point. Besides, it is difficult to incorporate v into the algorithm mainly because the setting of v requires the information of optimal solution at any time point. Algorithms have to know what kind of robust solutions in ROOT they are searching for, just as the distribution information of disturbances is informed to the algorithm in traditional robust optimization [1,10].

To the best of our knowledge, the only algorithm available for ROOT in the literature is from [7]. An algorithm framework which contains an optimizer, a database, an approximator and a predictor, was proposed in [7]. The basic idea is to average solution's fitness over the past and the future. To be more specific, the optimizer in the framework searches solutions based on a metric [2] which is the average over solution's previous, current and future fitnesses. Solution's previous fitness is approximated using previously evaluated solutions which are stored in the database, while solution's future fitness is predicted based on its previous and current fitnesses using the predictor. The construction of the framework is intuitively sensible for ROOT. However, the metric suffers from two main problems. Firstly, the metric does not incorporate the information

[1] Without loss of generality, we consider maximization problems in this paper.

[2] A metric is a function which assigns a scalar to a solution to differentiate good solutions from bad ones.

of robustness definitions. Therefore, the optimizer does not really know what kind of robust solutions it is searching for. Secondly, estimated fitnesses (either previous or future fitnesses) are used in the metric without any consideration of the accuracy of the estimator (approximator or predictor). This is inappropriate as reliable estimations should be favoured in the metric. For example, if two solutions have the same metric value, the one with more reliable estimation should be considered better than the other.

This paper thus tries to overcome the shortcomings mentioned above regarding existing work for ROOT by first developing two robustness definitions, namely *survival time* and *average fitness*, and a corresponding performance measurement for ROOT. New metrics, based on which *survival time* and *average fitness* are optimized respectively using population-based algorithms, are also proposed. Specially, our metrics incorporate the information of robustness definitions and take estimator's accuracy into consideration. The remainder of the paper is structured as follows. Section 2 presents the robustness definitions *survival time* and *average fitness* in ROOT. After that, one performance measurement is suggested comparing algorithm's ability in finding robust solutions in ROOT. The new metrics are then described in Section 3. Experimental results are reported in Section 4 with regard to performances of the old metric in [7] and our metrics on our performance measurement for ROOT. Finally, conclusions and future work are discussed in Section 5.

2 Robustness Definitions and Performance Measurement

A DOP is different from a static optimization problem only if the DOP is solved in an on-line manner [6,9], i.e., the algorithm for DOPs has to provide solutions repeatedly as time goes by. Suppose at time t, the algorithm comes up with a solution $\overrightarrow{X}_t$. The robustness of solution $\overrightarrow{X}_t$ can be defined as:

- the *survival time* F^s equal to the maximal time interval starting from time t during which the fitness of solution $\overrightarrow{X}_t$ stays above a pre-defined fitness threshold δ:

$$F^s(\overrightarrow{X}, t, \delta) = max\{0 \cup \{l | F_i(\overrightarrow{X}) \geq \delta,\ \forall i,\ t \leq i \leq t + l\}\}, \tag{1}$$

- or alternatively the *average fitness* F^a over a pre-defined time window T starting from time t:

$$F^a(\overrightarrow{X}, t, T) = \frac{1}{T} \sum_{i=t}^{t+T-1} F_i(\overrightarrow{X}). \tag{2}$$

Both robustness definitions (*survival time* F^s and *average fitness* F^a) do not require the information of optimal solution at any time point, and thus are not restricted to academic studies. For *survival time* F^s, the fitness threshold δ places a constraint on solution's current fitness, which is not satisfied in robustness

definitions used in [7]. More importantly, our robustness definitions have user-defined parameters (fitness threshold δ and time window T), which makes it easy to incorporate them into algorithms.

We would like to make a clear distinction between robustness definitions of solutions in ROOT and performance measurements for ROOT algorithms. As a DOP should be solved in an on-line manner and algorithms have to provide solutions repeatedly, algorithms should not be compared just at one time point but across the whole time period. As we consider discrete-time DOPs in this paper, a DOP can be represented as a sequence of static fitness functions $(F_1, F_2, ..., F_N)$ during a considered time interval $[t_0, t_{end})$. Given the robustness definitions in Equation 1 and 2, we could define ROOT performance measurement for time interval $[t_0, t_{end})$ as follows:

$$Performance^{ROOT} = \frac{1}{N}\sum_{i=1}^{N} E(i), \tag{3}$$

where $E(i)$ is the robustness (either *survival time* F^s or *average fitness* F^a) of the solution determined by the algorithm during the time of F_i.

It should be noted that performance measurement for ROOT proposed here is dependent on parameter settings, being either δ if *survival time* F^s is investigated or T if *average fitness* F^a is considered. Therefore, in order to compare algorithms' ROOT abilities comprehensively, results should be reported under different settings of δ or T.

3 New Metrics for Finding Robust Solutions in ROOT

A metric for finding robust solutions in ROOT was proposed in [7], which takes the form $\sum_{i=t-p}^{t+q} F_i(\overrightarrow{X})$ when the current time is t, where p and q are two parameters to control how many time steps looking backward and forward respectively. As discussed in Section 1, the metric does not incorporate the information of robustness definition, and the estimation accuracy is not taken into consideration. To address the two problems, we propose new metrics in the following. As our new metrics take robustness definitions into consideration, we describe the new metrics in the context of *survival time* F^s and *average fitness* F^a respectively.

3.1 Metric for Robustness Definition: *Survival Time*

If we restrict that the metric to optimize *survival time* F^s is a function of solution's current and future fitnesses and user-defined fitness threshold δ, we can define the metric $\hat{F}^s$ as follows:

$$\hat{F}^s(\overrightarrow{X}, t, \delta) = \begin{cases} F_t(\overrightarrow{X}) & \text{if } F_t(\overrightarrow{X}) < \delta, \\ \delta + w * \hat{l} & \text{otherwise,} \end{cases} \tag{4}$$

where $F_t(\overrightarrow{X})$ is the current fitness of solution $\overrightarrow{X}$, and $\hat{l}$ is used to represent the number of consecutive fitnesses which are no smaller than δ starting from

the beginning of the fitness sequence $(\hat{F}_{t+1}(\overrightarrow{X}), ..., \hat{F}_{t+L}(\overrightarrow{X}))$. $\hat{F}_{t+i}(\overrightarrow{X})$ is the predicted fitness of solution $\overrightarrow{X}$ at time $t+i$, $1 \leq i \leq L$. $\hat{l}$ can be seen as an explicit estimation of solution's *survival time* robustness. As a result, every time the metric $\hat{F}^s$ is calculated, L number of solution's future fitnesses are predicted if $F_t(\overrightarrow{X}) \geq \delta$. w is the weight coefficient associated with the accuracy of the estimator which is used to calculate $\hat{F}_{t+i}(\overrightarrow{X})$, $1 \leq i \leq L$. In this paper, the root mean square error R^{err} is employed as the accuracy measurement, which takes the form:

$$R^{err} = \sqrt{\frac{\sum_{i=1}^{n_t} e_i^2}{n_t}}, \tag{5}$$

where n_t is the number of sample data, and e_i is the absolute difference between the estimated value produced by the estimator and the true value for the ith sample data. In order to make sure that a larger weight is assigned when the corresponding estimator is considered more accurate, w takes an exponential function of R^{err}:

$$w = exp(-\theta * R^{err}), \tag{6}$$

where θ is a control parameter, $\theta \in [0, +\infty)$. The design of metric $\hat{F}^s$ is reasonable in the sense that it takes the form of current fitness if the current fitness is below the fitness threshold δ. On the other hand, if the current fitness is no smaller than δ, $\hat{F}^s$ only depends on $w * \hat{l}$ which is the product of the weight coefficient w and solution's *survival time* robustness estimation $\hat{l}$.

3.2 Metric for Robustness Definition: *Average Fitness*

The design of a metric for optimizing *average fitness* F^a is more straightforward than that for *survival time* F^s. Basically, in order to estimate *average fitness* F^a, solution's future fitnesses are predicted first and then summed together with solution's current fitness. Therefore, if the user-defined time window is T and the current time is t, we have the following metric:

$$\hat{F}^a(\overrightarrow{X}, t, T) = F_t(\overrightarrow{X}) + \sum_{i=1}^{T-1} (\hat{F}_{t+i}(\overrightarrow{X}) - \theta * R^{err}), \tag{7}$$

where $\hat{F}_{t+i}(\overrightarrow{X})$, θ and R^{err} take the same meaning as those used for the metric $\hat{F}^s$.

With the new metrics developed in Equation 4 and 7, we can have our new algorithms for ROOT by incorporating them into the generic population-based algorithm framework developed in [7]. For more details of the framework, readers can refer to [7].

4 Experimental Study

We conduct two groups of experiments in this section. The objective of the first group is to demonstrate that it is necessary to incorporate the robustness definitions into the algorithm for ROOT. The metric in [7] (denoted as Jin's metric)

is compared with our metrics, denoted as *survival time* metric for Equation 4 and *average fitness* metric for Equation 7. One true previous fitness and four future predicted fitnesses are used for Jin's metric, the setting of which is reported to have the best performance in [7]. Five future fitnesses are predicted ($L = 5$) for the *survival time* metric when the robustness definition is *survival time*. The control parameter θ is set to be 0 in the first group, which means the accuracy of the estimator is not considered temporarily. In the second group, our metrics are investigated with the control parameter θ set to be 0 and 1. The aim is to demonstrate the advantage of making use of estimator's accuracy when calculating the metrics.

4.1 Experimental Setup

Test Problem. All experiments in this paper are conducted on the modified Moving Peaks Benchmark (mMPB). mMPB is derived from Branke's Moving Peaks Benchmark (MPB) [3] by allowing each peak having its own change severities. The reason to modify MPB that way is to make some parts of the landscape change more severely than other parts. Basically, mMPB consists of several peak functions whose height, width and center position change over time. The mMPB can be described as:

$$F_t(\overrightarrow{X}) = \max_{i=1}^{i=m}\{H_t^i - W_t^i * ||\overrightarrow{X} - \overrightarrow{C}_t^i||_2\}, \tag{8}$$

where H_t^i, W_t^i and C_t^i denote the height, width and center of the *ith* peak function at time t, $\overrightarrow{X}$ is the design variable, and m is the total number of peaks. Besides, the timer t adds 1 after a certain period of time Δe which is measured by the number of fitness evaluations. H_t^i, W_t^i and $\overrightarrow{C}_t^i$ change as follows:

$$\begin{aligned}
H_{t+1}^i &= H_t^i + height_severity^i * N(0,1),\\
W_{t+1}^i &= W_t^i + width_severity^i * N(0,1),\\
\overrightarrow{C}_{t+1}^i &= \overrightarrow{C}_t^i + \overrightarrow{v}_{t+1}^i,\\
\overrightarrow{v}_{t+1}^i &= \frac{s * ((1-\lambda) * \overrightarrow{r} + \lambda * \overrightarrow{v}_t^i)}{\parallel (1-\lambda) * \overrightarrow{r} + \lambda * \overrightarrow{v}_t^i \parallel},
\end{aligned} \tag{9}$$

where N(0,1) denotes a random number drawn from Gaussian distribution with zero mean and variance one. Each peak's height H_t^i and width W_t^i vary according to its own $height\_severity^i$ and $width\_severity^i$, which are randomly initialized within $height\_severity\_range$ and $width\_severity\_range$ respectively. H_t^i and W_t^i are constrained in the range $[30, 70]$ and $[1, 12]$ respectively. The center $\overrightarrow{C}_t^i$ is moved by a vector $\overrightarrow{v}^i$ of length s in a random direction ($\lambda = 0$) or a direction exhibiting a trend ($\lambda > 0$). The random vector $\overrightarrow{r}$ is created by drawing random numbers in $[-0.5, 0.5]$ for each dimension and then normalizing its length to s. The settings of mMPB are summarized in Table 1.

In our experiments, we generate 150 consecutive fitness functions with a fixed random number generator. All the results presented are based on 30 independent runs of algorithms with different random seeds.

Table 1. Parameter settings of the mMPB benchmark

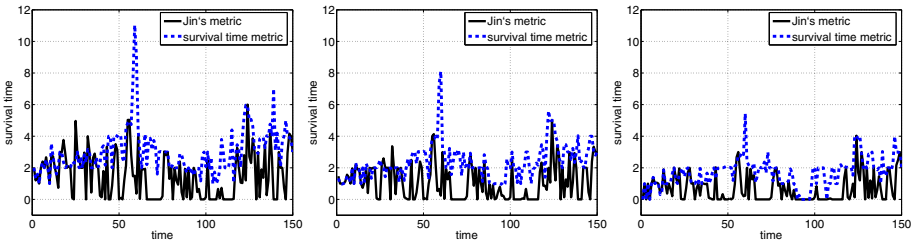
number of peaks, m	5
change frequency, Δe	2500
number of dimensions, D	2
search range	$[0, 50]$
height range	$[30, 70]$
initial height	50
width range	$[1, 12]$
initial width	6
$height\_severity\_range$	$[1, 10]$
$width\_severity\_range$	$[0.1, 1]$
trend parameter, λ	1
scale parameter, s	1

Parameter Settings. We adopt a simple PSO algorithm as the optimizer in this paper. The PSO algorithm used in this paper takes the constriction version. For details of the PSO algorithm, readers are advised to refer to [4]. The swarm population size is 50. The constants c_1 and c_2, which are used to bias particle's attraction to local best and global best, are both set to be 2.05, and therefore the constriction factor χ takes a value 0.729844. The velocity of particles are constricted within the range $[-V_{MAX}, V_{MAX}]$. The value of V_{MAX} is set to be the upper bounds of the search range, which is 50 in our case.

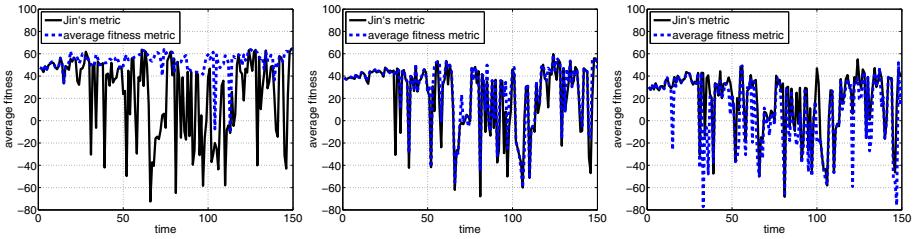
We use the Autoregressive (AR) model for the prediction task. An AR model of order ψ takes the form $Y_t = \epsilon + \sum_{i=1}^{\psi} \eta_i * Y_{t-i}$ where ϵ is the white noise and Y_t is the time series data at time t. We use the least square method to estimate AR model parameters $\overrightarrow{\eta}$ ($\overrightarrow{\eta} = (\eta_1, \eta_2, ...\eta_\psi)$). The parameter ψ is set to be 5 and the latest time series of length 15 are used as the training data. If AR model accuracy is considered, the first 12 time steps are chosen as the training data, and the latest 3 time steps are used to calculate R^{err}. We omit the process of approximating solution's previous fitness but use solution's true previous fitness for both Jin's metric and our metrics. The reasons are we would like to exclude the effects of approximation error but focus on the effects of prediction error on the metrics, and also it is relatively easy to approximate solution's previous fitness given enough historical data, which is usually available in population-based algorithms.

4.2 Simulation Results

The results of the first group experiment are plotted in Fig. 1. In Fig. 1(a), (b), (c) and (d), we can see that the results achieved by our metrics with $\theta = 0$ are generally above those achieved by Jin's metric. This is mainly because our metrics take the corresponding robustness definitions into consideration, and therefore are better at capturing user's preferences of robustness. Our metrics have similar results with Jin's in Fig. 1(e) and (f). This is because by setting T



(a) Fitness threshold $\delta = 40$ (b) Fitness threshold $\delta = 45$ (c) Fitness threshold $\delta = 50$



(d) Time window $T = 2$ (e) Time window $T = 4$ (f) Time window $T = 6$

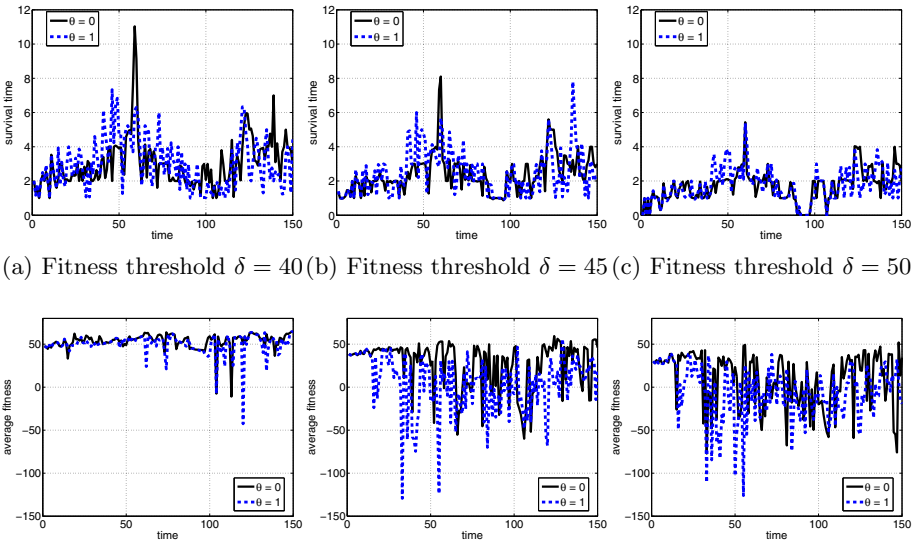
Fig. 1. The averaged robustness over 30 runs for each time step, produced by Jin's metric and our metrics (θ is set to be 0) under robustness definitions of *survival time* F^s and *average fitness* F^a with different settings of δ and T respectively

equal to 4 or 6, our metrics happen to have similar forms to Jin's metric. All these results are further summarized in Table 2.

The results of the second group experiment are plotted in Fig. 2. The advantage of incorporating estimator's accuracy into metrics has been confirmed in results for *survival time* F^s. This may due to the fact that R^{err} is in accordance with the accuracy in calculating *survival time* estimation $\hat{l}$. However, we can see a performance degrade in making use of estimator's accuracy in the results for *average fitness* F^a. This means R^{err} may not be a good indicator of estimator's accuracy in predicting solution's future fitness. All these results are further summarized in Table 2.

Table 2. Performance measurement in Equation 3 of investigated algorithms (standard deviation in bracket). Wilcoxon rank sum tests at a 0.05 significance level are conducted between every two of the three algorithms. Significance is indicated in **boldness** for the first and the second, star $*$ for the second and the third and underline for the first and the third.

Algorithms	$\delta = 40$	$\delta = 45$	$\delta = 50$	$T = 2$	$T = 4$	$T = 6$
Jin's	1.53(0.08)	1.11(0.06)	0.69(0.05)	25.32(1.20)	22.20(1.08)	**18.46**(1.06)
Ours ($\theta = 0$)	**3.02**(0.05)	**2.39**(0.05)	**1.69**(0.03)	**53.48***(0.38)	**26.99***(1.12)	8.82*(1.11)
Ours ($\theta = 1$)	3.01(0.08)	2.49*(0.05)	1.72*(0.04)	50.15(0.64)	4.91(1.81)	-5.26(1.98)



(a) Fitness threshold $\delta = 40$ (b) Fitness threshold $\delta = 45$ (c) Fitness threshold $\delta = 50$

(d) Time window $T = 2$ (e) Time window $T = 4$ (f) Time window $T = 6$

Fig. 2. The averaged robustness over 30 runs for each time step, produced by our metrics when θ is set to be 0 and 1 under robustness definitions of *survival time* F^s and *average fitness* F^a with different settings of δ and T respectively

5 Conclusions and Future Work

In this paper, we pointed out the inappropriateness of existing robustness definitions in ROOT and developed two new definitions *survival time* F^s and *average fitness* F^a. Moreover, we developed two novel metrics (*survival time* metric and *average fitness* metric) based on which population-based algorithms search for robust solutions in ROOT. In contrast with the metric in [7], our metrics not only take robustness definitions into consideration but also make use of estimator's accuracy.

From the simulation results, we can arrive that it is necessary to incorporate the information of robustness definitions into the algorithm for ROOT. In other words, the algorithm has to know what kind of robust solutions it is searching for. Secondly, estimator's accuracy can have a large influence on algorithm's performance, and it is important to develop appropriate accuracy measure considering the robustness to be maximized in ROOT. Also, it is equally important to employ estimator's accuracy information in guiding the search for robust solutions in ROOT.

For the future work, the variance of solution's future fitnesses can be considered as a second objective, and existing multi-objective algorithms can be adapted for it. Also, in what way estimation models should interact with search algorithms is still an open question in ROOT, as solution's future fitnesses are considered in ROOT and prediction task is inevitable.

Acknowledgment. This work was support by Honda Research Institute Europe, EU FP7 (grant No. 247619), and National Natural Science Foundation of China (grant Nos. 61028009, U0835002, and 61175065).

References

1. Beyer, H.G., Sendhoff, B.: Robust optimization-a comprehensive survey. Computer Methods in Applied Mechanics and Engineering 196(33-34), 3190–3218 (2007)
2. Blackwell, T., Branke, J., Li, X.: Particle swarms for dynamic optimization problems. In: Swarm Intelligence, pp. 193–217 (2008)
3. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 3, IEEE (1999)
4. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
5. Fu, H., Sendhoff, B., Tang, K., Yao, X.: Characterizing environmental changes in robust optimization over time. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2012)
6. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
7. Jin, Y., Tang, K., Yu, X., Sendhoff, B., Yao, X.: A framework for finding robust optimal solutions over time. In: Memetic Computing, pp. 1–16 (2012)
8. Li, C., Yang, S.: A general framework of multipopulation methods with clustering in undetectable dynamic environments. IEEE Transactions on Evolutionary Computation 16(4), 556–577 (2012)
9. Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation 6, 1–24 (2012)
10. Paenke, I., Branke, J., Jin, Y.: Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. IEEE Transactions on Evolutionary Computation 10(4), 405–420 (2006)
11. Rohlfshagen, P., Yao, X.: Dynamic combinatorial optimisation problems: an analysis of the subset sum problem. Soft Computing 15(9), 1723–1734 (2011)
12. Simões, A., Costa, E.: Prediction in evolutionary algorithms for dynamic environments using markov chains and nonlinear regression. In: Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation, pp. 883–890. ACM Press, New York (2009)
13. Yang, S., Jin, Y., Ong, Y.S.: Evolutionary Computation in Dynamic and Uncertain Environments. Springer, Heidelberg (2007)
14. Yu, X., Jin, Y., Tang, K., Yao, X.: Robust optimization over time–A new perspective on dynamic optimization problems. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–6. IEEE (2010)

An Ant-Based Selection Hyper-heuristic for Dynamic Environments

Berna Kiraz[1], A. Şima Etaner-Uyar[2], and Ender Özcan[3]

[1] Institute of Science and Technology, Istanbul Technical University, Turkey
berna.kiraz@marmara.edu.tr
[2] Department of Computer Engineering, Istanbul Technical University, Turkey
etaner@itu.edu.tr
[3] School of Computer Science, University of Nottingham, UK
ender.ozcan@nottingham.ac.uk

Abstract. Dynamic environment problems require adaptive solution methodologies which can deal with the changes in the environment during the solution process for a given problem. A selection hyper-heuristic manages a set of low level heuristics (operators) and decides which one to apply at each iterative step. Recent studies show that selection hyper-heuristic methodologies are indeed suitable for solving dynamic environment problems with their ability of tracking the change dynamics in a given environment. The choice function based selection hyper-heuristic is reported to be the best hyper-heuristic on a set of benchmark problems. In this study, we investigate the performance of a new learning hyper-heuristic and its variants which are inspired from the ant colony optimization algorithm components. The proposed hyper-heuristic maintains a matrix of pheromone intensities (utility values) between all pairs of low level heuristics. A heuristic is selected based on the utility values between the previously invoked heuristic and each heuristic from the set of low level heuristics. The ant-based hyper-heuristic performs better than the choice function and even its improved version across a variety of dynamic environments produced by the Moving Peaks Benchmark generator.

1 Introduction

Many real world constraint optimization problems contain a set of components which might change in time separately or concurrently. Some of these components include the problem instance, the objectives and the constraints. A *good* solution method needs to be adaptive and intelligent to be able to deal with the complexities introduced by such a dynamic environment and track the changes. Branke [2] categorized these changes based on (i) *frequency*, (ii) *severity*, (iii) *predictability* of a change, and (iv) *cycle length/cycle accuracy* which is a property defining somewhat periodicity and precision of changes. There is a variety of approaches dealing with different types of changes in literature. Most of the

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 626–635, 2013.
© Springer-Verlag Berlin Heidelberg 2013

approaches handling dynamic environment problems are modified from the existing approaches designed for solving static problems based on different strategies. More details on dynamic environments can be found in [2,7,13,19].

Hyper-heuristics are emerging methodologies which explore the space of heuristics to solve hard computational problems [5,3]. In this study, we use *selection hyper-heuristic* methodologies which combine two main components, namely heuristic selection and move acceptance methods for solving dynamic environment problems [4,15]. A selection hyper-heuristic controls a set of low level heuristics, choosing the most appropriate one to apply to a solution in hand and deciding to accept or reject the newly created solution at each step. There are learning hyper-heuristic methodologies as well as the ones which do not utilize any learning at all. An online learning selection hyper-heuristic can incorporate learning either into the heuristic selection, or move acceptance methods. These components attempt to improve performance during the search process. If learning takes place during heuristic selection, frequently, a mechanism is used to score the performance of each low level heuristic. Then a heuristic is chosen based on these scores. Nareyek [14] tested *Reinforcement Learning* (RL) heuristic selection on Orc Quest and modified logistics domain. Initially, scores are initialised to the same value for all heuristics, e.g., 0. After a chosen heuristic is applied to a solution, if there is improvement, then its score is increased; otherwise it is decreased, e.g. by one. A heuristic is chosen using different mechanisms and the best strategy appears to be selecting the heuristic with maximum score. Cowling et al. [6] investigated the performance of different heuristic selection methods on a real world scheduling problem. One of them is a learning method referred to as *Choice Function* (CF) which maintains a score for each low level heuristic taking a weighted average of three values; how well it performs individually and as a successor of the previously invoked heuristic and the elapsed time since its last call. Then the heuristic with the maximum score is selected and applied to the current solution at a given step. Drake and Özcan [9] proposed a modified version of CF improving its performance (ICF) in which weights dynamically change, enforcing the search process to go into diversification faster than usual, when the successive moves are non-improving.

There is strong empirical evidence that selection hyper-heuristics work for not only discrete combinatorial problems [3] but also discrete and continuous dynamic environment problems [11,12,18,17], being able to respond to the changes in such an environment rapidly. In this study, we describe a new learning hyper-heuristic for dynamic environments, which is designed based on the ant colony optimization algorithm components. The proposed hyper-heuristic maintains a matrix of pheromone intensities (utility values) between all pairs of low level heuristics. A heuristic is selected based on the utility values between the previously invoked heuristic and each heuristic from the set of low level heuristics. We investigate the performance of the proposed hyper-heuristic controlling a set of parameterised mutation operators for solving the dynamic environment problems produced by the Moving Peaks Benchmark (MPB) generator.

2 Proposed Method

In this study, we propose a selection hyper-heuristic incorporating a novel heuristic selection method, called the *Ant based Selection* (*AbS*), which is based on a simple ant colony optimization (ACO) algorithm [8]. Most of the mechanisms used in ACO are adapted within *AbS*. A distinct feature of *AbS* is that unlike ACO, *AbS* is based on a single point based search framework. In most of the selection hyper-heuristics, there is a *heuristic selection* step followed by an *acceptance* step. After the heuristic selection step using *AbS*, we employ the generic Improving-and-Equal acceptance scheme, which accepts a new solution of better or equal quality as compared to the previous solution.

Similar to the Choice Function and Reinforcement Learning heuristic selection schemes, *AbS* also incorporates an online learning mechanism using a matrix of utility values. In *AbS*, each heuristic pair is associated with a pheromone trail value (τ_{h_i,h_j}) which shows the desirability of selecting the j^{th} heuristic after the application of the i^{th} heuristic. All pheromone trails are initialized with a small value τ_0. *AbS* selects a random low-level heuristic at the first step. In the following steps, the most appropriate low-level heuristic is selected and applied to a solution in hand based on the pheromone trail information.

AbS consists of heuristic selection and pheromone update stages. For the first stage, we consider two variants of heuristic selection schemes. In both variants, the heuristic h_b with the highest pheromone trail ($h_b = \max_{i=1..n} \tau_{h_c,h_j}$) is selected with a probability of q_0 where h_c is the previously selected heuristic and n is the number of low-level heuristics. Otherwise, methods inspired by two of the mate selection techniques most commonly used in Evolutionary Algorithms [10] are employed to determine the next heuristic to select. In the first variant, like in ACO, the next heuristic is determined based on probabilities proportional to the pheromone levels of each heuristic pair. This is similar to the roulette wheel mate selection in Evolutionary Algorithms. In this method, *AbSrw* selects the next heuristic h_s with a probability which is proportional to the pheromone trail value of τ_{h_c,h_s}. However, in the second variant (*AbSts*), the choice of the next heuristic is based on tournament selection. *AbSts* chooses the next heuristic h_s with the highest pheromone trail ($h_s = \max_{i=1..k} \tau_{h_c,h_j}$ where k is the tournament size).

After selecting a heuristic, pheromone trails are modified. Firstly, pheromone values on the pheromone matrix are decreased by a constant factor (evaporation) for all pairs of heuristics as given in Equation 1.

$$\tau_{h_i,h_j} = (1-\rho)\tau_{h_i,h_j} \tag{1}$$

where $0 < \rho \leq 1$ is the pheromone evaporation rate.

After evaporation, the pheromone value between only h_c and h_s (τ_{h_c,h_s}) is increased using Equation 2.

$$\tau_{h_c,h_s} = \tau_{h_c,h_s} + \Delta\tau \tag{2}$$

where h_c is the previously selected heuristic and h_s is the last selected heuristic. $\Delta\tau$ is the amount of pheromone added and is defined as in Equation 3.

$$\Delta\tau = 1/f_c \tag{3}$$

where f_c is the fitness value of the new solution generated by applying the selected heuristic h_s.

3 Experimental Design

In this study, we perform experiments with our new hyper-heuristic for dynamic environments, combining the Ant-based selection scheme and the Improving-and-Equal acceptance technique. For comparison, we also experiment with previously used selection mechanisms which incorporate some form of online learning and are shown to be successful in dynamic environments [12], namely the Choice Function (CF) and Reinforcement Learning (RL). In this paper, we also include an improved version of the Choice Function (ICF) proposed in [9]. These selection mechanisms are also used together with the Improving-and-Equal acceptance technique.

In the experiments, we used the Moving Peaks Benchmark (MPB) generator [1] to generate the various dynamic environments. In MPB, the height, width and location of the peaks forming the multimodal and multidimensional landscape are changed every Δe iterations by adding a normally distributed random variable to the heights and the widths of the peaks and by shifting their locations with a vector of fixed length in a random direction. In some applications, a time-invariant basis function is also included in the generated landscapes. The *height_severity*, the *width_severity* and *vlength* parameters determine the severity of the changes in the heights, the widths and the locations of the peaks respectively. For the fixed parameters of the MPB in all the experiments, we used the settings taken from [1,2] as follows: cone peak function, 5 peaks, 5 dimensions, range of peak heights $\in [30, 70]$, range of peak widths $\in [0.8, 7.0]$ and range of values in each dimension $\in [0.0, 100.0]$. We did not use a basis function and allowed no correlation between consecutive movements of a peak. For the Δe, *height_severity*, the *width_severity* and *vlength* settings we used the ones given in [12] and labeled as EXPSET2. Based on these settings, Δe is taken as 6006 fitness evaluations for low frequency (LF), 1001 for medium frequency (MF) and 126 for high frequency (HF); the *height_severity*, the *width_severity* and *vlength* parameters are taken as given in Table 1 which correspond to low severity (LS), medium severity (MS) and high severity (HS) changes.

Table 1. MPB parameter settings for high, medium and low severity changes

Setting	LS	MS	HS
vlength	1.0	5.0	10.0
height_severity	1.0	5.0	10.0
width_severity	0.1	0.5	1.0

A real-valued vector corresponds to the coordinates of a point in the search space generated by the MPB. The fitness of a candidate solution at a given time t is given by its *error*, which is calculated as its distance to the optimum in terms

of the objective function value at time t. Therefore the problem becomes that of minimising the *error* values.

The search algorithm searches through the landscape by perturbing these candidate solutions at each step to obtain a new one using a parameterized Gaussian mutation, $N(0, \sigma^2)$, where σ denotes the standard deviation. We used the same settings for the mutation operators as in [12], which are implemented as seven different standard deviations; {0.5, 2, 7, 15, 20, 25, 30}. These mutation operators are used as the low-level heuristics in the hyper-heuristic framework.

The parameters of the proposed Ant-based selection scheme are chosen as follows: ρ is set to 0.1. Each entry in the pheromone matrix is initialized to $\tau_0 = 1/f_s$ where f_s is the fitness value of initial solution. For *AbSrw*, we experiment with seven q_0 values: $\{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. For *AbSts*, we consider five tournament size values: $k = \{2, 3, 4, 5, 6\}$ as well as the above given seven q_0 values. We also included a second version of the approach where $\Delta\tau$ is calculated in a different way so that pheromone values decrease more gradually. In this case, $\Delta\tau$ is calculated as $\Delta\tau = 0.1 * (1/f_c)$. *AbSrw* with slow decreasing and *AbSts* with slow decreasing are denoted as *sAbSrw* and *sAbSts*, respectively.

For the parameter settings of the other heuristic selection methods, their proposed settings from literature are used. In Reinforcement Learning, the scores of all heuristics are initialized to 15 with lower and upper bounds as 0 and 30 respectively as given in [16]. At each step, the score of a heuristic that improves performance is increased by 1 and otherwise it is decreased by 1. In Choice Function, α, β, and δ are initialized to 0.5 with updates of ± 0.01 at each iteration as given in [9]. In the Improved Choice Function, ϕ, which refers α and β, and δ are initialized to 0.5. If the heuristic improves performance, the values of ϕ are set to 0.99. Otherwise, the values of ϕ_t at time t are calculated as $\phi_t = \max\{\phi_{t-1} - 0.01, 0.01\}$. In addition, δ is calculated as $\delta_t = 1 - \phi_t$.

We assume that all programs are made aware when a change in the environment occurs. For the Reinforcement Learning, Choice Function and the Improved Choice Function selection methods, when a change occurs, the current solution is re-evaluated. For the proposed Ant-based selection scheme, this is not required. The parameters of none of the heuristic selection methods are reset when the environment changes. Due to the nature of the acceptance mechanism, Improving-and-Equal, the first solution candidate generated after each environment change is accepted regardless of its solution quality.

For evaluating the performance of the approaches, we used the offline error [2] metric. The error of a candidate solution at a given time is calculated as its distance to the optimum in terms of the objective function value at that time. The offline error is calculated as a cumulative average of the errors of the best candidate solutions found so far since the last change. At the end of a run, a lower overall offline error value is desired indicating a *good* performance.

Each run is repeated 100 times for each setting where 20 changes occur, i.e. there are 21 consecutive stationary periods. This means that there are $maxIterations = (NoOfChanges + 1) * ChangePeriod$ number of iterations per run. For analysis of statistical significance of the differences obtained

between the results of various approaches, we performed ANOVA tests together with Tukey's HSD at a confidence level of 95%.

4 Results and Discussion

In this section we provide the results of the experiments and their discussions. Table 2 shows the results of the q_0 tests for both *AbSrw* and *sAbSrw*. In the table $q_0 = 0.0$ means that the next heuristic is chosen using only the roulette-wheel selection. However, $q_0 = 1.0$ means that roulette wheel selection is not used and always the heuristic with the best score (pheromone value) is chosen to be applied. The results show that there are no statistically significant differences between most cases, however, the best values are provided by different q_0 values for different frequency-severity pairs. Therefore, to avoid overtuning, we decided to choose a setting which provided an acceptable performance in most of the cases for both approaches.For the rest of the experiments we continued with a setting of $q_0 = 0.5$ for both *AbSrw* and *sAbSrw*.

Table 2. Final offline error results of various q_0 settings for *AbSrw* and *sAbSrw* under the tested change frequency-severity pairs

Algorithm	q_0	LF			MF			HF		
		LS	MS	HS	LS	MS	HS	LS	MS	HS
AbSrw	0.0	3.58	7.66	9.88	4.65	8.86	12.32	9.89	17.03	24.04
	0.1	3.46	7.64	10.28	4.38	9.34	11.75	10.14	17.14	25.24
	0.3	3.93	8.03	10.24	4.61	8.77	12.35	9.29	16.45	24.61
	0.5	3.72	8.11	10.58	4.64	9.60	13.05	9.40	15.46	24.26
	0.7	4.19	8.43	10.56	4.98	10.84	13.18	10.43	17.42	25.49
	0.9	3.93	10.42	10.85	4.78	10.46	13.65	12.50	19.81	28.94
	1.0	3.96	10.37	12.33	5.53	10.85	14.00	15.90	23.06	31.92
sAbSrw	0.0	3.73	7.45	9.73	4.39	8.53	11.94	10.35	17.11	25.59
	0.1	3.72	7.09	9.88	4.49	8.51	12.33	9.95	17.18	24.37
	0.3	3.98	7.60	10.19	4.39	8.25	12.47	8.88	15.44	24.03
	0.5	3.64	7.40	10.99	4.18	8.70	12.42	7.93	14.94	22.90
	0.7	3.88	8.58	10.95	4.48	9.49	12.53	8.91	16.00	24.13
	0.9	4.28	8.83	11.94	4.40	10.10	13.33	8.69	15.43	23.71
	1.00	4.88	9.96	12.56	6.53	12.62	14.80	13.54	19.35	25.19

Then, we performed experiments to set the q_0 and tournament size values for the *AbSts* and *sAbSts* variations. The experimental results which are not provided in here due to space limitation revealed that for these variants of the approach, the best setting of these two parameters depends highly on the dynamics of the environment, i.e. the change frequency and severity values. Since we observed that the settings of these two parameters are sensitive, we decided to develop an adaptive mechanism as future work. For this study, we chose a simpler approach. For those cases where tournament selection is to be applied, each time we let

the tournament size be determined randomly with equal probability from among the five pre-determined tournament size levels. We performed the q_0 analysis for *AbSts* and *sAbSts* based on this scheme. Table 3 shows the final offline error results for various q_0 settings for *AbSts* and *sAbSts* when the tournament sizes are determined randomly. We chose $q_0 = 0.1$ for *AbSts* and $q_0 = 0.9$ for *sAbSts*, since each approach delivers an acceptable performance in most of the cases with these settings which are used for the rest of the experiments.

Table 3. Final offline error results of various q_0 settings for *AbSts* and *sAbSts* with random tournament sizes under the tested change frequency-severity pairs

Algorithm	q_0	LF			MF			HF		
		LS	MS	HS	LS	MS	HS	LS	MS	HS
AbSts	0.0	3.84	8.52	10.90	5.24	10.23	13.03	13.43	18.36	25.46
	0.1	4.06	8.02	10.94	5.07	9.57	12.85	13.49	19.04	25.26
	0.3	3.74	8.29	11.11	5.17	9.97	13.19	13.12	19.33	25.33
	0.5	3.93	8.09	11.88	5.01	10.20	13.36	12.84	18.35	25.95
	0.7	3.92	8.32	11.78	5.15	9.69	13.18	12.87	18.83	26.59
	0.9	4.03	8.78	12.01	4.92	11.96	12.96	13.86	21.46	30.14
	1.0	3.99	9.98	11.74	5.46	10.79	14.62	14.64	22.26	30.59
sAbSts	0.0	4.15	8.19	10.37	5.05	10.11	13.00	12.24	17.25	24.22
	0.1	4.24	8.49	10.72	5.22	10.10	13.05	12.80	18.56	24.72
	0.3	3.68	8.37	11.95	4.92	10.45	12.44	11.38	17.89	24.15
	0.5	3.91	8.43	11.01	4.66	9.84	13.04	10.10	16.74	24.88
	0.7	4.08	8.63	11.51	4.64	10.31	13.19	9.68	16.93	23.91
	0.9	4.03	9.00	11.74	4.64	10.03	12.39	9.12	16.76	23.42
	1.0	4.75	10.57	12.60	7.70	12.25	14.56	13.40	18.13	24.27

Finally, we compare our approaches with those obtained using the heuristic selection methods taken from literature, namely RL, CF and ICF. Table 4 shows the results of these comparisons. The better results are marked in bold in the table. As can be seen, RL and ICF are worse than the others for all cases and these differences are statistically significant, with ICF being the worst of all.

ICF aims to emphasize the intensification component of the generic CF by automatically increasing the weight of relevant components as soon as there is improvement. Diversification, on the other hand, is introduced at a gradually increasing rate. This property works in solving stationary combinatorial optimization problems as shown in [9], but not in dynamic environments. The changes in the environment mislead ICF and it gets worse than CF in all cases. It was shown in [11,12] that CF outperforms RL for all tested change dynamics using the MPB. Therefore, the poor performance of RL in the current experiments is also to be expected.

The results show that all versions of the proposed heuristic selection scheme provide better results than CF, except for the LF-LS and MF-LS cases, however, the results are close. Among the versions of the proposed heuristic selection scheme, *sAbSrw* provides the better results in most cases.

Table 4. Final offline error results for the proposed heuristic selection schemes and RL, CF and ICF. Here, for both *AbSrw* and *sAbSrw* $q_0 = 0.5$, for *AbSts* $q_0 = 0.1$ and for *sAbSts* $q_0 = 0.9$ with random tournament size settings.

Algorithm	LF			MF			HF		
	LS	MS	HS	LS	MS	HS	LS	MS	HS
AbSrw	3.72	8.11	**10.58**	4.64	9.60	13.05	9.40	15.46	24.26
sAbSrw	3.64	**7.40**	10.99	4.18	**8.70**	12.42	**7.93**	**14.94**	**22.90**
AbSts	4.06	8.02	10.94	5.07	9.57	12.85	13.49	19.04	25.26
sAbSts	4.03	9.00	11.74	4.64	10.03	**12.39**	9.12	16.76	23.42
CF	**3.52**	9.80	11.88	**4.09**	10.95	13.60	7.98	15.67	24.39
ICF	10.07	15.29	18.14	11.04	19.69	20.76	19.12	27.29	32.79
RL	4.43	9.05	12.18	5.97	12.44	14.54	10.29	15.69	24.57

Due to lack of space, we cannot provide all the statistical comparison tables. Table 5 provides a summary. In the table the values under $s+$ show the total number of times the corresponding approach was statistically significantly better than the others and $s-$ counts show the vice versa. $\geq$ counts show the number of times the corresponding approach was better than the others based on average results, however, the differences were not statistically significant. $\leq$ counts show the vice versa. The results in the table are ordered in decreasing order of the $s+$ counts. The counts in the table show that *sAbSrw* has the most $s+$ and $\geq$ counts with CF following close behind.

Table 5. Summary of statistical significance comparisons, where $s+$ is the total number of times the corresponding approach was statistically significantly better than the others, $s-$ is the vice versa, $\geq$ is the number of times the corresponding approach was better than the others based on average results (no statistical significance), and $\leq$ is the vice versa.

Algorithm	$s+$	$s-$	$\geq$	$\leq$
sAbSrw	15	0	34	5
CF	13	1	18	22
AbSrw	12	0	24	18
RL	11	7	3	33
AbSts	10	9	16	19
sAbSts	10	0	23	21
ICF	0	54	0	0

As can be seen, *sAbSrw* is better than CF in many cases, but in several others, their performance is comparable. However, the most important issue is the fact that *sAbSrw* (and also all the other proposed variants) are more suitable to be used in dynamic environments than RL, CF and ICF because the proposed heuristic selection schemes do not require any special actions to be performed

when the environment changes, whereas for the others, right after an environment change, the last solution candidate in the previous environment needs to be re-evaluated. This is a drawback for two reasons: it makes change detection necessary and it also wastes fitness evaluations, especially in environments where the change frequencies are very high.

5 Conclusion

In this paper, we proposed a new heuristic selection scheme for selection hyper-heuristics, especially for use in dynamic environments. In previous studies [11,12,18], existing heuristic selection mechanisms were tested in various types of dynamic environments and those that incorporated some form of online learning were shown to be successful. One drawback of these approaches for dynamic environments is that they require the re-evaluation of the last candidate solution in the previous environment for score calculation. As well as wasting computing resources for the re-evaluation, this also means that the algorithm needs to detect when the environment changes. The proposed heuristic selection does not require any special actions when the environment changes. Moreover, the empirical results show that the proposed heuristic selection scheme provides slightly better performance than the heuristic selection scheme previously reported to be the best in dynamic environments. The results are very encouraging and promote further study.

A drawback of the proposed method is that it has a number of parameters and the performance of the proposed heuristic selection method is sensitive to the initial setting of those parameters for some cases. As future work, we plan to develop adaptive mechanisms to alleviate the need for parameter tuning and enhance the performance of the proposed approach even further. Additionally, our future work will focus on the investigation of move acceptance methods which are more suitable for dynamic environments as selection hyper-heuristic components.

Acknowledgements. B. Kiraz is supported by TÜBİTAK 2211-National Scholarship Program for PhD students. The study is supported in part by EPSRC, grant EP/F033214/1 (The LANCS Initiative Postdoctoral Training Scheme).

References

1. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Congress on Evolutionary Computation, CEC 1999, vol. 3, pp. 1875–1882. IEEE (1999)
2. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer (2002)
3. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Rong, Q.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society (to appear)

4. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, International Series in Operations Research and Management Science, pp. 449–468. Springer (2010)
5. Chakhlevitch, K., Cowling, P.: Hyperheuristics: Recent developments. In: Cotta, C., Sevaux, M., Sirensen, K. (eds.) Adaptive and Multilevel Metaheuristics. SCI, pp. 3–29. Springer, Heidelberg (2008)
6. Cowling, P.I., Kendall, G., Soubeiga, E.: A Hyperheuristic Approach to Scheduling a Sales Summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
7. Cruz, C., Gonzalez, J., Pelta, D.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft Computing - A Fusion of Foundations, Methodologies and Applications 15, 1427–1448 (2011)
8. Dorigo, M., Stützle, T.: Ant Colony Optimizations. MIT Press (2004)
9. Drake, J.H., Özcan, E., Burke, E.K.: An Improved Choice Function Heuristic Selection for Cross Domain Heuristic Search. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 307–316. Springer, Heidelberg (2012)
10. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)
11. Kiraz, B., Uyar, A.Ş., Özcan, E.: An Investigation of Selection Hyper-heuristics in Dynamic Environments. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A.I., Merelo, J.J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 314–323. Springer, Heidelberg (2011)
12. Kiraz, B., Uyar, A.S., Özcan, E.: Selection hyper-heuristics in dynamic environments. Journal of the Operational Research Society
13. Morrison, R.W.: Designing evolutionary algorithms for dynamic environments. Springer (2004)
14. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: Metaheuristics: Computer Decision-Making, pp. 523–544. Kluwer Academic Publishers (2001)
15. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12, 3–23 (2008)
16. Özcan, E., Misir, M., Ochoa, G., Burke, E.K.: A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing 1(1), 39–59 (2010)
17. Özcan, E., Uyar, Ş., Burke, A.,, E.: A greedy hyper-heuristic in dynamic environments. In: GECCO 2009 Workshop on Automated Heuristic Design: Crossing the Chasm for Search Methods, pp. 2201–2204 (2009)
18. Uludağ, G., Kiraz, B., Etaner-Uyar, A.Ş., Özcan, E.: A Framework to Hybridize PBIL and a Hyper-heuristic for Dynamic Environments. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 358–367. Springer, Heidelberg (2012)
19. Yang, S., Ong, Y.S., Jin, Y. (eds.): Evolutionary Computation in Dynamic and Uncertain Environments. SCI. Springer (2007)

Author Index