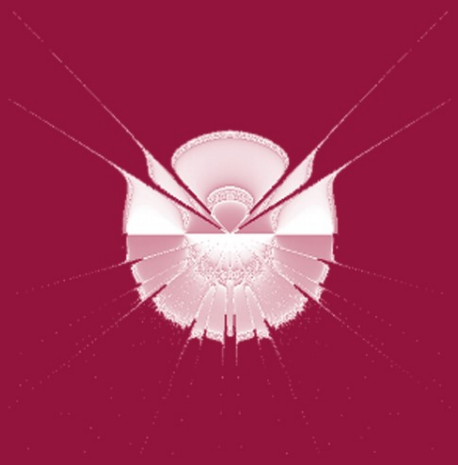
Cecilia Di Chio et al. (Eds.)

Applications of Evolutionary Computation

LNCS 6024

EvoApplications 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC
Istanbul, Turkey, April 2010, Proceedings, Part I



Springer

Lecture Notes in Computer Science 6024

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Cecilia Di Chio Stefano Cagnoni
Carlos Cotta Marc Ebner Anikó Ekárt
Anna I. Esparcia-Alcázar Chi-Keong Goh
Juan J. Merelo Ferrante Neri Mike Preuss
Julian Togelius Georgios N. Yannakakis (Eds.)

Applications of Evolutionary Computation

EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC
Istanbul, Turkey, April 7-9, 2010
Proceedings, Part I

Springer

Volume Editors

see next page

Cover illustration:
"Pelegrina Galathea" by Stayko Chalakov (2009) Aston University, UK

Library of Congress Control Number: 2010923234

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

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

ISSN 0302-9743
ISBN-10 3-642-12238-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-12238-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Volume Editors

Cecilia Di Chio
Dept. of Mathematics and Statistics
University of Strathclyde, UK
cecilia@stams.strath.ac.uk

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

Carlos Cotta
Departamento Lenguajes y Ciencias
de la Computación
University of Malága, Spain
ccottap@lcc.uma.es

Marc Ebner
Wilhelm-Schickard-Institut
für Informatik
Universität Tübingen, Germany
marc.ebner@wsii.uni-tuebingen.de

Anikó Ekárt
Knowledge Engineering Research
Group, Aston University
Birmingham, UK
ekarta@aston.ac.uk

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

Chi-Keong Goh
Advanced Technology Centre
Rolls-Royce
Singapore
chi.keong.goh@rolls-royce.com

Juan J. Merelo
Departamento de Electrónica y
Tecnología de los Computadores
Universidad de Granada, Spain
jmerelo@geneura.ugr.es

Ferrante Neri
Department of Mathematical
Information Technology
University of Jyväskylä, Finland
ferrante.neri@jyu.fi

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

Julian Togelius
Center for Computer Games Research
IT University of Copenhagen,
Denmark
julian@togelius.com

Georgios N. Yannakakis
Center for Computer Games Research
IT University of Copenhagen,
Denmark
yannakakis@itu.dk

Preface

Evolutionary Computation (EC) techniques are efficient, nature-inspired methods based on the principles of natural evolution and genetics. Due to their efficiency and simple underlying principles, these methods can be used for a diverse range of activities including problem solving, optimization, machine learning and pattern recognition. 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 2010, which included a range of events on application-oriented aspects of EC. Since 1998, EvoApplications — formerly known as EvoWorkshops — has provided a unique opportunity for EC researchers to meet and discuss application aspects of EC and has been an important link between EC research and its application in a variety of domains. During these 12 years, new events have arisen, some have disappeared, while others have matured to become conferences of their own, such as EuroGP in 2000, EvoCOP in 2004, and EvoBIO in 2007. And from this year, EvoApplications has become a conference as well.

EvoApplications is part of EVO*, Europe's premier co-located events in the field of evolutionary computing. EVO* was held from the 7th to the 9th of April 2010 in the beautiful city of Istanbul, Turkey, which was European City of Culture in 2010. Evo* 2010 included, in addition to EvoApplications, EuroGP, the main European event dedicated to genetic programming; EvoCOP, the main European conference on EC in combinatorial optimization; EvoBIO, the main European conference on EC and related techniques in bioinformatics and computational biology. The proceedings for all of these events, EuroGP 2010, EvoCOP 2010 and EvoBIO 2010, are also available in the LNCS series (volumes 6021, 6022, and 6023).

Moreover, thanks to the large number of submissions received, the proceedings for EvoApplications 2010 are divided across two volumes. The present volume, which contains contributions for: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC; and volume two (LNCS 6025), which contains contributions for: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoMUSART, and EvoTRANSLOG.

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 2010 consisted of the following individual events:

- *EvoCOMNET*, the 7^{th} European Event on the Application of Nature-Inspired Techniques for Telecommunication Networks and other Parallel and Distributed Systems
- *EvoCOMPLEX*, the 1^{st} European Event on Evolutionary Algorithms and Complex Systems
- *EvoENVIRONMENT*, the 2^{nd} European Event on Nature-Inspired Methods for Environmental Issues
- *EvoFIN*, the 4^{th} European Event on Evolutionary and Natural Computation in Finance and Economics
- *EvoGAMES*, the 2^{nd} European Event on Bio-inspired Algorithms in Games
- *EvoIASP*, the 12^{th} European Event on Evolutionary Computation in Image Analysis and Signal Processing
- *EvoINTELLIGENCE*, the 1^{st} European Event on Nature-Inspired Methods for Intelligent Systems
- *EvoMUSART*, the 8^{th} European Event on Evolutionary and Biologically Inspired Music, Sound, Art and Design
- *EvoNUM*, the 3^{rd} European Event on Bio-inspired Algorithms for Continuous Parameter Optimization
- *EvoSTOC*, the 7^{th} European Event on Evolutionary Algorithms in Stochastic and Dynamic Environments
- *EvoTRANSLOG*, the 4^{th} European Event on Evolutionary Computation in Transportation and Logistics

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

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

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

EvoGAMES aims to focus the scientific developments onto computational intelligence techniques that may be of practical value for utilization in existing or future games. Recently, games, and especially video games, have become an important commercial factor within the software industry, providing an excellent test bed for the application of a wide range of computational intelligence methods.

EvoIASP, the longest-running of all EvoApplications which celebrated its 12th edition this year, has been the first international event solely dedicated to the applications of EC to image analysis and signal processing in complex domains of high industrial and social relevance.

EvoINTELLIGENCE is devoted to the use of nature-inspired methods to create all kinds of intelligent systems. The scope of the event includes evolutionary robotics, artificial life and related areas. Intelligent systems do not necessarily have to exhibit human or animal-like intelligence. Intelligent behavior can also be found in everyday devices such as a digital video recorder or handheld devices such as an MP3 player which learn from the human who is operating the device.

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

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

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

EvoTRANSLOG deals with all aspects of the use of evolutionary computation, local search and other nature-inspired optimization and design techniques for the transportation and logistics domain. The impact of these problems on the modern economy and society has been growing steadily over the last few decades, and the event aims at design and optimization techniques such as

evolutionary computing approaches allowing the use of computer systems for systematic design, optimization, and improvement of systems in the transportation and logistics domain.

Continuing in the tradition of adapting the list of the events to the needs and demands of the researchers working in the field of evolutionary computing, EvoINTERACTION, the European Event on Interactive Evolution and Humanized Computational Intelligence, and EvoHOT, the European Event on Bio-inspired Heuristics for Design Automation, decided not to run in 2010 and will run again in 2011. Two new events were also proposed this year: EvoCOMPLEX, the First European Event on Evolutionary Algorithms and Complex Systems, and EvoINTELLIGENCE, the First European Event on Nature-Inspired Methods for Intelligent Systems.

The number of submissions to EvoApplications 2010 was once again very high, cumulating 188 entries (with respect to 133 in 2008 and 143 in 2009). The following table shows relevant statistics for EvoApplications 2010 (both short and long papers are considered in the acceptance statistics), compared with those from the 2009 edition:

Event	2010			2009		
	Submissions	Accept	Ratio	Submissions	Accept	Ratio
EvoCOMNET	17	12	71%	21	15	71%
EvoCOMPLEX	12	6	50%	-	-	-
EvoENVIRONMENT	5	4	80%	5	4	80%
EvoFIN	17	10	59%	14	8	57%
EvoGAMES	25	15	60%	15	10	67%
EvoIASP	24	15	62%	14	7	50%
EvoINTELLIGENCE	8	5	62%	-	-	-
EvoMUSART	36	16	44%	26	17	65%
EvoNUM	25	15	60%	16	9	56%
EvoSTOC	11	6	54%	11	7	64%
EvoTRANSLOG	11	5	45%	11	6	54%
Total	191	109	57%	143	91	64%

As for previous years, accepted papers were split into oral presentations and posters. However, this year, each event made their own decision on paper length for these two categories. Hence, for some events, papers in both categories are of the same length. The acceptance rate of 57.1% for EvoApplications 2010, along with the significant number of submissions, is an indicator of the high quality of the articles presented at the events, showing the liveliness of the scientific movement in the corresponding fields.

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

- Computer Engineering Department of Istanbul Technical University, Turkey, for supporting the local organization
- Istanbul Technical University, Microsoft Turkey, and the Scientific and Technological Research Council of Turkey, for their patronage of the event

– Centre for Emergent Computing at Edinburgh Napier University, Scotland, for administrative help and event coordination

We want to especially acknowledge our invited speakers: Kevin Warwick (University of Reading, UK), Luigi Luca Cavalli-Sforza (Stanford School of Medicine, USA); and Günther Raidl (Vienna University of Technology, Austria) and Jens Gottlieb (SAP, Walldorf, Germany) for their special EvoCOP 10th anniversary talk.

We are also very grateful to all the people who provided local support, in particular Sanem Sarıel-Talay, Şule Gündüz-Öğüdücü, Ayşegül Yayımlı, Gülşen Cebiroğlu-Eryiğit, and H. Turgut Uyar.

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

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

April 2010

Cecilia Di Chio
Stefano Cagnoni
Carlos Cotta
Marc Ebner
Anikó Ekárt
Anna I. Esparcia-Alcázar
Chi-Keong Goh
Juan J. Merelo
Ferrante Neri
Mike Preuss
Julian Togelius
Georgios N. Yannakakis

Organization

EvoApplications 2010 was part of EVO* 2010, Europe's premier co-located events in the field of evolutionary computing, that also included the conferences EuroGP 2010, EvoCOP 2010, and EvoBIO 2010.

Organizing Committee

EvoApplications Chair:	Cecilia Di Chio, University of Strathclyde, UK
Local Chairs:	A. Şima (Etaner) Uyar, Istanbul Technical University, Turkey
Publicity Chair:	Stephen Dignum, University of Essex, UK
EvoCOMNET Co-chairs:	Gianni A. Di Caro, IDSIA, Switzerland Muddassar Farooq, National University of Computer and Emerging Sciences, Pakistan Ernesto Tarantino, Institute for High Performance Computing and Networking, Italy
EvoCOMPLEX Co-chairs:	Carlos Cotta, University of Malaga, Spain Juan J. Merelo, University of Granada, Spain
EvoENVIRONMENT Co-chairs:	Marc Ebner, University of Tübingen, Germany Neil Urquhart, Edinburgh Napier University, UK
EvoFIN Co-chairs:	Anthony Brabazon, University College Dublin, Ireland Michael O'Neill, University College Dublin, Ireland
EvoGAMES Co-chairs:	Mike Preuss, TU Dortmund University, Germany Julian Togelius, IT University of Copenhagen, Denmark Georgios N. Yannakakis, IT University of Copenhagen, Denmark
EvoIASP Chair:	Stefano Cagnoni, University of Parma, Italy

EvoINTELLIGENCE Co-chairs:	Marc Ebner, University of Tübingen, Germany Cecilia Di Chio, University of Strathclyde, UK
EvoMUSART Co-chairs:	Penousal Machado, University of Coimbra, Portugal Gary Greenfield, University of Richmond, USA
EvoNUM Co-chairs:	Anna Isabel Esparcia-Alcazar, ITI - Universidad Politécnica de Valencia, Spain Anikó Ekárt, Aston University, UK
EvoSTOC Co-chairs:	Ferrante Neri, University of Jyväskylä, Finland Chi-Keong Goh, Advanced Technology Centre Rolls-Royce, Singapore
EvoTRANSLOG Co-chairs:	Andreas Fink, Helmut-Schmidt-University Hamburg, Germany Jörn Grahl, Johannes Gutenberg University, Germany

Program Committees

EvoCOMNET Program Committee

Özgür B. Akan	Middle East Technical University, Turkey
Enrique Alba	University of Malaga, Spain
Qing Anyong	National University of Singapore, Singapore
Payman Arabshahi	University of Washington, USA
Mehmet E. Aydin	University of Bedfordshire, UK
Iacopo Carreras	CREATE-NET, Italy
Arindam K. Das	University of Washington, USA
Falko Dressler	University of Erlangen, Germany
Frederick Ducatelle	IDSIA, Switzerland
Luca Gambardella	IDSIA, Switzerland
Jin-Kao Hao	University of Angers, France
Malcolm I. Heywood	Dalhousie University, Canada
Byrant Julstrom	St. Cloud State University, USA
Graham Kendall	University of Nottingham, UK
Kenji Leibnitz	Osaka University, Japan
Manuel Lozano-Marquez	University of Granada, Spain
Domenico Maisto	ICAR CNR, Italy
Ronaldo Menezes	Florida Institute of Technology, USA
Martin Middendorf	University of Leipzig, Germany
Roberto Montemanni	IDSIA, Switzerland
Chien-Chung Shen	University of Delaware, USA
Tony White	Carleton University, Canada
Lidia Yamamoto	University of Basel, Switzerland
Nur Zincir-Heywood	Dalhousie University, Canada

EvoCOMPLEX Program Committee

Antonio Córdoba	Universidad de Sevilla, Spain
Carlos Cotta	Universidad de Málaga, Spain
Jordi Delgado	Universitat Politècnica de Catalunya, Spain
Carlos Gershenson	UNAM, Mexico
Mario Giacobini	Università di Torino, Italy
Anca Gog	Babes-Bolyai University, Romania
Márk Jelasity	University of Szeged, Hungary
Juan Luis Jiménez	University of Granada, Spain
Jose Fernando Mendes	Universidade de Aveiro, Portugal
Juan J. Merelo	Universidad de Granada, Spain
Joshua L. Payne	University of Vermont, USA
Mike Preuss	Universität Dortmund, Germany
Katya Rodríguez-Vázquez	UNAM, Mexico
Kepa Ruiz-Mirazo	Euskal Herriko Unibertsitatea, Spain
Luciano Sánchez	Universidad de Oviedo, Spain
Robert Schaefer	AGH University of Science and Technology, Poland
Marco Tomassini	Université de Lausanne, Switzerland
Fernando Tricas	Universidad de Zaragoza, Spain
Sergi Valverde	Universitat Pompeu Frabra, Spain
Leonardo Vanneschi	University of Milano-Bicocca, Italy

EvoENVIRONMENT Program Committee

Stefano Cagnoni	University of Parma, Italy
Pierre Collet	Université de Strasbourg, France
Kevin Cullinane	Edinburgh Napier University, UK
Marc Ebner	Universität Tübingen, Germany
James A Foster	University of Idaho, USA
Nanlin Jin	University of Leeds, UK
Rhyd Lewis	Cardiff University, UK
William Magette	University College Dublin, Ireland
R I (Bob) McKay	Seoul National University, Korea
Michael O'Neill	University College Dublin, Ireland
Stefano Pizzuti	Energy New Tech. and Environment Agency, Italy
Tom Rye	Edinburgh Napier University, UK
Carlo Santulli	University of Rome "La Sapienza", Italy
Marc Schoenauer	INRIA, France
Terence Soule	University of Idaho, USA
John Summerscales	University of Plymouth, UK
Neil Urquhart	Edinburgh Napier University, UK
Tina Yu	Memorial University of Newfoundland, Canada
Mengjie Zhang	University of Wellington, New Zealand

EvoFIN Program Committee

Eva Alfaro-Cid	Instituto Tecnológico de Informática, Spain
Antonia Azzini	Università degli Studi di Milano, Italy
Anthony Brabazon	University College Dublin, Ireland
Louis Charbonneau	Concordia University, Canada
Gregory Connor	National University of Ireland Maynooth, Ireland
Ian Dempsey	Pipeline Trading, USA
Rafal Drezewski	AGH University of Science and Technology, Poland
Manfred Gilli	University of Geneva and Swiss Finance Institute, Switzerland
Philip Hamill	University of Ulster, UK
Ronald Hochreiter	WU Vienna University of Economics and Business, Austria
Youwei Li	Queen's University Belfast, UK
Dietmar Maringer	University of Basel, Switzerland
Michael O'Neill	University College Dublin, Ireland
Philip Saks	University of Essex, UK
Robert Schafer	AGH University of Science and Technology, Poland
Andrea Tettamanzi	Università Degli Studi di Milano, Italy
Garnett Wilson	Memorial University of Newfoundland, Canada

EvoGAMES Program Committee

Lourdes Araujo	UNED, Spain
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Luigi Barone	University of Western Australia, Australia
Simon Colton	Imperial College London, UK
Ernesto Costa	Universidade de Coimbra, Portugal
Carlos Cotta	Universidad de Málaga, Spain
Marc Ebner	University of Tübingen, Germany
Anikó Ekárt	Aston University, UK
Anna Esparcia Alcázar	Instituto Tecnológico de Informática, Spain
Francisco Fernández	Universidad de Extremadura, Spain
Antonio J Fernández Leiva	Universidad de Málaga, Spain
Mario Giacobini	Università degli Studi di Torino, Italy
Johan Hagelbäck	Blekinge Tekniska Högskola, Sweden
John Hallam	University of Southern Denmark, Denmark
David Hart	Fall Line Studio, USA
Philip Hingston	Edith Cowan University, Australia
Stefan Johansson	Blekinge Tekniska Högskola, Sweden
Rilla Khaled	IT University of Copenhagen, Denmark

Elias Kosmatopoulos	Dimocritian University of Thrace, Greece
Krzysztof Krawiec	Poznan University of Technology, Poland
Pier Luca Lanzi	Politecnico di Milano, Italy
Simon Lucas	University of Essex, UK
Penousal Machado	Universidade de Coimbra, Portugal
Juan J. Merelo	Universidad de Granada, Spain
Risto Miikkulainen	University of Texas at Austin, USA
Antonio Mora	Universidad de Granada, Spain
Mike Preuss	Universität Dortmund, Germany
Steffen Priesterjahn	University of Paderborn, Germany
Moshe Sipper	Ben-Gurion University, Israel
Terence Soule	University of Idaho, USA
Julian Togelius	IT University of Copenhagen, Denmark
Georgios N. Yannakakis	IT University of Copenhagen, Denmark

EvoIASP Program Committee

Antonia Azzini	University of Milan-Crema, Italy
Lucia Ballerini	University of Edinburgh, UK
Leonardo Bocchi	University of Florence, Italy
Stefano Cagnoni	University of Parma, Italy
Oscar Cordon	European Center for Soft Computing, Spain
Sergio Damas	European Center for Soft Computing, Spain
Ivanoe De Falco	ICAR - CNR, Italy
Antonio Della Cioppa	University of Salerno, Italy
Laura Dipietro	MIT, USA
Marc Ebner	University of Tübingen, Germany
Francesco Fontanella	University of Cassino, Italy
Şpela Ivekoviç	University of Dundee, UK
Mario Koeppen	Kyushu Institute of Technology, Japan
Krisztof Krawiec	Poznan University of Technology, Poland
Jean Louchet	INRIA, France
Evelyne Lutton	INRIA, France
Luca Mussi	University of Parma, Italy
Ferrante Neri	University of Jyväskylä, Finland
Gustavo Olague	CICESE, Mexico
Riccardo Poli	University of Essex, UK
Stephen Smith	University of York, UK
Giovanni Squillero	Politecnico di Torino, Italy
Kiyoshi Tanaka	Shinshu University, Japan
Andy Tyrrell	University of York, UK
Leonardo Vanneschi	University of Milan Bicocca, Italy
Mengjie Zhang	Victoria University of Wellington, New Zealand

EvoINTELLIGENCE Program Committee

Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Peter Bentley	University College London, UK
Stefano Cagnoni	University of Parma, Italy
Cecilia Di Chio	University of Strathclyde, UK
Marc Ebner	Eberhard Karls Universität Tübingen, Germany
Mario Giacobini	University of Turin, Italy
Greg Hornby	University of California Santa Cruz, USA
Christian Jacob	University of Calgary, Canada
Gul Muhammad Kahn	University of Engineering and Technology, Pakistan
Gabriela Kokai	Fraunhofer Inst. für Integrated Circuits, Germany
William B. Langdon	King's College, London, UK
Penousal Machado	University of Coimbra, Portugal
Julian Miller	University of York, UK
Gustavo Olague	CICESE, Mexico
Michael O'Neill	University College Dublin, Ireland
Thomas Ray	University of Oklahoma, USA
Marc Schoenauer	INRIA, France
Moshe Sipper	Ben-Gurion University, Israel
Ivan Tanev	Doshisha University, Japan
Mengjie Zhang	Victoria University of Wellington, New Zealand

EvoMUSART Program Committee

Mauro Annunziato	Plancton Art Studio, Italy
Peter Bentley	University College London, UK
Eleonora Bilotta	University of Calabria, Italy
Tim Blackwell	Goldsmiths College, University of London, UK
Simon Colton	Imperial College, UK
Oliver Bown	Monash University, Australia
Paul Brown	University of Sussex, UK
Stefano Cagnoni	University of Parma, Italy
Amilcar Cardoso	University of Coimbra, Portugal
Vic Ciesielski	RMIT, Australia
Palle Dahlstedt	Göteborg University, Sweden
Hans Dehlinger	Independent Artist, Germany
Steve DiPaola	Simon Fraser University, Canada
Alan Dorin	Monash University, Australia
Erwin Driessens	Independent Artist, The Netherlands
Philip Galanter	Texas A&M College of Architecture, USA
Pablo Gervás	Universidad Complutense de Madrid, Spain

Andrew Gildfind	Google, Inc., Australia
Carlos Grilo	Instituto Politécnico de Leiria, Portugal
David Hart	Independent Artist, USA
Amy K. Hoover	University of Central Florida, USA
Andrew Horner	University of Science & Technology, Hong Kong
Christian Jacob	University of Calgary, Canada
Colin Johnson	University of Kent, UK
Craig Kaplan	University of Waterloo, Canada
Matthew Lewis	Ohio State University, USA
Alain Lioret	Paris 8 University, France
Bill Manaris	College of Charleston, USA
Ruli Manurung	University of Indonesia, Indonesia
Jonatas Manzolli	UNICAMP, Brazil
Jon McCormack	Monash University, Australia
James McDermott	University of Limerick, Ireland
Eduardo Miranda	University of Plymouth, UK
Nicolas Monmarché	University of Tours, France
Gary Nelson	Oberlin College, USA
Luigi Pagliarini	PEAM, Italy & University of Southern, Denmark
Rui Pedro Paiva	University of Coimbra, Portugal
Alejandro Pazos	University of A Coruna, Spain
Somnuk Phon-Amnuaisuk	Multimedia University, Malaysia
Rafael Ramirez	Pompeu Fabra University, Spain
Juan Romero	University of A Coruna, Spain
Brian Ross	Brock University, Canada
Artemis Sanchez Moroni	Renato Archer Research Center, Brazil
Antonino Santos	University of A Coruna, Spain
Kenneth O. Stanley	University of Central Florida, USA
Jorge Tavares	University of Coimbra, Portugal
Stephen Todd	IBM, UK
Paulo Urbano	Universidade de Lisboa, Portugal
Anna Ursyn	University of Northern Colorado, USA
Maria Verstappen	Independent Artist, The Netherlands
Gerhard Widmer	Johannes Kepler University Linz, Austria

EvoNUM Program Committee

Eva Alfaro-Cid	ITI – Universidad Politécnica de Valencia, Spain
Anne Auger	INRIA, France
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Xavier Blasco	Universidad Politécnica de Valencia, Spain
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria

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

EvoSTOC Program Committee

Hussein Abbass	University of New South Wales, Australia
Dirk Arnold	Dalhousie University, Canada
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Peter Bosman	Centre for Mathematics and Computer Science, The Netherlands
Juergen Branke	University of Karlsruhe, Germany
Andrea Caponio	Technical University of Bari, Italy
Ernesto Costa	University of Coimbra, Portugal
Kalyanmoy Deb	Indian Institute of Technology Kanpur, India
Andries Engelbrecht	University of Pretoria, South Africa
Yaochu Jin	Honda Research Institute Europe, Germany
Anna V. Kononova	University of Leeds, UK
Jouni Lampinen	University of Vaasa, Finland
Xiaodong Li	RMIT University, Australia
John McCall	Robert Gordon University, UK

Ernesto Mininno	University of Jyväskylä, Finland
Yew Soon Ong	Nanyang Technological University of Singapore, Singapore
Zhang Qingfu	University of Essex, UK
William Rand	University of Maryland, USA
Khaled Rasheed	University of Georgia, USA
Hendrik Richter	University of Leipzig, Germany
Philipp Rohlfshagen	University of Birmingham, UK
Kay Chen Tan	National University of Singapore, Singapore
Ke Tang	University of Science and Technology of China, China
Yoel Tenne	Sydney University, Australia
Renato Tinos	Universidade de Sao Paulo, Brazil
Ville Tirronen	University of Jyväskylä, Finland
Shengxiang Yang	University of Leicester, UK
Gary Yen	Oklahoma State University, USA

EvoTRANSLOG Program Committee

Christian Blum	Univ. Politecnica Catalunya, Spain
Peter A.N. Bosman	Centre for Mathematics and Computer Science, The Netherlands
Marco Caserta	University of Hamburg, Germany
Loukas Dimitriou	National Technical University of Athens, Greece
Karl Doerner	University of Vienna, Austria
Martin Josef Geiger	Helmut-Schmidt-University Hamburg, Germany
Stefan Irnich	RWTH Aachen University, Germany
Hoong Chuin Lau	Singapore Management University, Singapore
Christian Prins	University of Technology of Troyes, France
Franz Rothlauf	University of Mainz, Germany
Kay Chen Tan	National University of Singapore, Singapore
Theodore Tsekeris	Center of Planning and Economic Research, Greece
Stefan Voß	University of Hamburg, Germany
Oliver Wendt	University of Kaiserslautern, Germany

Sponsoring Institutions

- Istanbul Technical University, Istanbul, Turkey
- Microsoft Turkey
- Scientific and Technological Research Council of Turkey
- The Centre for Emergent Computing at Edinburgh Napier University, Scotland

Table of Contents – Part I

EvoCOMPLEX Contributions

EvoGAMES Contributions

EvoIASP Contributions

EvoINTELLIGENCE Contributions

EvoNUM Contributions

EvoSTOC Contributions

Table of Contents – Part II

EvoCOMNET Contributions

EvoENVIRONMENT Contributions

EvoFIN Contributions

EvoMUSART Contributions

EvoTRANSLOG Contributions

Coevolutionary Dynamics of Interacting Species

Marc Ebner[1], Richard A. Watson[2], and Jason Alexander[3]

[1] Eberhard Karls Universität Tübingen, WSI für Informatik, Abt. RA, Sand 1, 72076 Tübingen, Germany
marc.ebner@wsii.uni-tuebingen.de
[2] University of Southampton, School of Electronics and Computer Science, Highfield, Southampton, SO17 1BJ, UK
raw@ecs.soton.ac.uk
[3] London School of Economics, Department of Philosophy, Logic and Scientific Method, Houghton Street, London WC2A 2AE, UK
J.Alexander@lse.ac.uk

Abstract. One of the open questions in evolutionary computation is how an arms race may be initiated between coevolving entities such that the entities acquire new behaviors and increase in complexity over time. It would be highly desirable to establish the necessary and sufficient conditions which lead to such an arms race. We investigate what these conditions may be using a model of competitive coevolution. Coevolving species are modeled as points which are placed randomly on a two-dimensional fitness landscape. The position of the species has an impact on the fitness landscape surrounding the phenotype of the species. Each species deforms the fitness landscape locally. This deformation, however, is not immediate. It follows the species after some latency period. We model evolution as a simple hill climbing process. Selection causes the species to climb towards the nearest optimum. We investigate the impact different conditions have on the evolutionary dynamics of this process. We will see that some conditions lead to cyclic or stationary behavior while others lead to an arms race. We will also see spontaneous occurrence of speciation on the two-dimensional landscape.

1 Introduction

Most research in evolutionary algorithms assumes that a population of individuals is adapting to a fixed fitness landscape as defined by the fitness function. However, some researchers also have looked at the problem of a population of individuals adapting to a fitness function which changes over time, e.g [5,9,16]. In nature, the environment to which species adapt over the course of time is never static. It is constantly changing because it mostly consists of other individuals which themselves undergo evolution. In our research, we are focusing on coevolving species and their evolutionary dynamics. We try to isolate important factors which have an impact on the coevolutionary dynamics of different species.

Let us assume that we have two species adapting to their local environment. The fitness landscapes of coevolving species are coupled. If one species achieves

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 1–10, 2010.

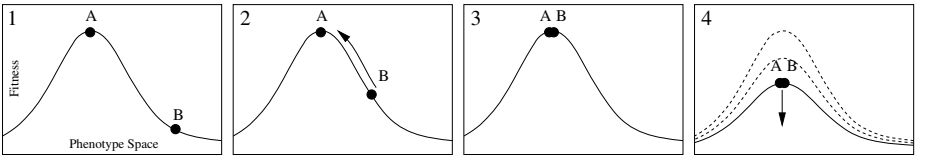


Fig. 1. Two species are coupled by a shared fitness landscape

a higher fitness, i.e. higher reproductive rate, this also has a significant impact on the fitness of the second species [7]. Two coevolving species may well reach a higher optimum than either one alone. This may be the result of an arms race which is induced between the two species [1]. An important question in this respect is: what are the necessary and sufficient conditions for an arms race between two or more coevolving species. For this work, we will speak of an arms race when two or more species try to out-compete each other.

In our simulations, the species are placed on a single landscape to which they have to adapt. The behavior of two species A and B is illustrated in Figure 1. Each species is driven towards a local optimum due to the selective pressure. Species B is moving towards the same optimum, i.e. is located inside the same niche. It is assumed that if the niche is occupied by two or more species then the fitness of these species is reduced.

Kauffman [7] suggested an abstract model for coevolution in order to investigate coevolutionary dynamics. In his model, epistatic links between genes have an influence on the smoothness or ruggedness of the landscape. Each species had a separate landscape which was coupled to other species. According to Kauffman, one of the conditions necessary for his coevolving system to accumulate beneficial traits and to maintain fitness at a high level is that the couplings within each species is roughly equal to the product of the coupling between species and the number of other species each species interacts with. In contrast to Kauffman's work, our model uses a single shared fitness landscape for all species.

Other authors have also looked at evolutionary dynamics in several different problem domains. For instance, Spector and Klein [12] visualized evolutionary dynamics of a set of agents each representing a particular species inside a three-dimensional environment. Observed behaviors included genetic drift, as well as emergence of collective and multicellularity. In their work, interaction does not occur through an explicit fitness landscape. Evolutionary dynamics in the context of biological games are described by Nowak and Sigmund [10]. Game dynamics of finite populations are considered by Taylor et al. [14]. Evolutionary dynamics on graphs are explored by Lieberman et al. [8].

Another focus is on the dynamics of coevolution. Can we provide an answer to the question what conditions lead to an arms race between two coevolving species and which don't? What conditions produce a stable attractor where no further progress is possible once the attractor has been reached? If we work with open ended evolutionary systems, e.g. self-reproducing programs [11], we don't want evolution to halt. Hence, it is important to understand which set of

conditions is necessary and sufficient in order to establish an arms race where the complexity of the coevolving species continuously improves.

2 Coevolutionary Dynamics on a Deformable Landscape

In order to visualize the dynamics of coevolution, we need an appropriate model. The model has to be simple enough such that computations can be carried out quickly. Hence, we have modeled evolution as a hill climbing process on a fitness landscape. This concept was originally introduced by Wright [18]. Modeling an entire population of individuals with reproduction and selection as would be found in the usual evolutionary model would be much more costly. Instead, we model only the population average phenotype of each species. Gradient ascent is then used to update this average. Thus, the evolutionary process consisting of replication, variation and selection is modeled as a single step. Each population is represented by an n-dimensional vector which describes its position in phenotype space. In the simplest case, i.e. a one-dimensional fitness landscape, we just have a scalar value. This scalar value describes the position of the species inside a one-dimensional world. Assuming that our world consists of n species then we require n scalar values. Experiments on a one-dimensional fitness landscape have been carried out by Ebner et al. [3]. In contrast to these experiments, we will be using a two-dimensional landscape. We will see that an interesting phenomena emerges as we switch from a one to a two-dimensional fitness landscape.

We experimented with three different models on how the population moves through phenotype space. The first method computes the sign of the gradient of the fitness landscape where the species is located and then moves the species one step to the right or left depending on the sign of this gradient. Let $f(x,t)$ be the height of the fitness landscape at time t. Let a species be located at position $x(t)$ then we compute the velocity $\dot{x}(t)$ using

$$\dot{x}(t) = \begin{cases} -1 & \text{if } \frac{\partial}{\partial x} f(x,t) < 0, \\ 0 & \text{if } \frac{\partial}{\partial x} f(x,t) = 0, \\ 1 & \text{if } \frac{\partial}{\partial x} f(x,t) > 0. \end{cases} \tag{1}$$

The second update method computes the gradient of the fitness landscape and then sets the velocity of the species to the gradient multiplied by a constant factor α. Whereas in equation (1) the movement of the population through phenotype space has constant speed (and fitness information merely controls the direction of movement), equation (2) presents a model where rate of change in phenotype is proportional to fitness gradient. This is more in alignment with a classical model where rate of change of fitness is proportional to fitness variance assuming that variance in phenotype is constant.

$$\dot{x}(t) = \alpha \frac{\partial}{\partial x} f(x,t) \tag{2}$$

The third update method also computes the gradient of the fitness landscape and integrates this gradient over time.

$$\dot{x}(t) = \alpha \left(\frac{\partial}{\partial x} f(x,t) \right) + \beta \dot{x}(t-1). \tag{3}$$

This equation suggests that the mean of the population moves in response to the gradient at the current position inside the fitness landscape and also in response to the gradient at the previous time step. The effect of the previous location of the population mean can be interpreted as an effect arising from overlapping generations of individuals i.e. not all members of the population are replaced every generation, some are maintained from the previous generation. Such momentum terms are known to have interesting effects on many kinds of dynamical systems, and the experiments that follow indicate that if such an effect is active in biological populations then they may have significant effects on coevolutionary dynamics.

The different species are placed on the same fitness landscape Figure 2(a). Each species deforms the landscape in its vicinity Figure 2(b). We use a Gaussian shaped deformation. The shape of the deformation can be considered to be due to the distribution of a population around the population average assuming Gaussian mutation. The species has a negative impact on fitness in its vicinity. Such a negative impact on fitness is also used in evolutionary algorithms (where it is called sharing [4]) when trying to populate different local optima. We use the same type of deformation for all species. The deformation of this landscape gets larger as more and more species occupy the same position.

The species are pushed towards local optima by the selective force of evolution Figure 2(c). The deformation caused by the species can either have an immediate or a latent effect on the local fitness. A latent effect only becomes apparent after some time steps Figure 2(d-f). We observe quite interesting behavior if the deformation caused by the species occurs after a latency period of several time

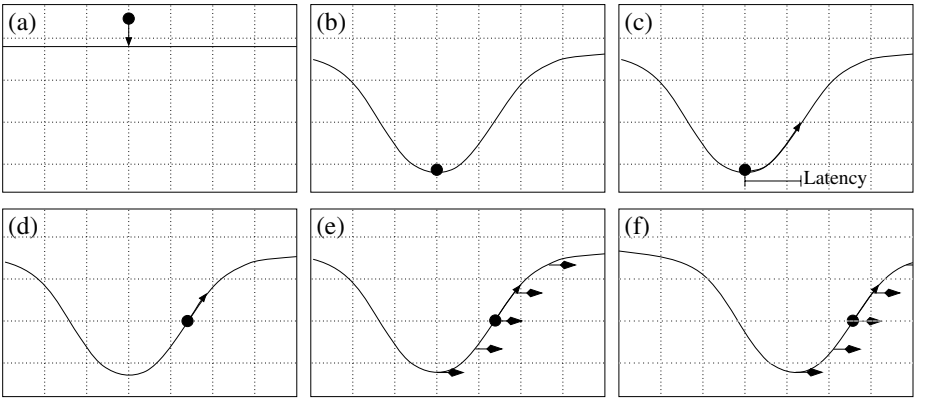


Fig. 2. Deformation of the fitness landscape

steps. The analog with natural evolution would be the excessive use of resources which are depleted after some time. Thus, we have basically two different modes. In the non-latent model, the deformation of the landscape is positioned wherever the species is located inside the fitness landscape. In the latent model, the deformation is positioned wherever the species was located some number of time steps in the past. In this model, the species may climb toward a local optimum. However, after some time, this local optimum is depressed by the presence of the species and the species then has to adapt to a new optimum. We set the latency to a constant value for all species. Once we use such a latency period, we will observe the Red Queen effect [13,17] as shown in Figure 2(e-f). The species tries to climb to the top of a local optima. However, the deformation follows the species. It thus seems as if no progress has been achieved. The species still has the same fitness even though it moved through phenotype space.

In summary, our model has the following parameters. Each species is described by its position on the landscape, an update rule, which describes how the position of the species for the next time step is derived, a function describing the deformation caused by the species, and an update method which describes how the deformation follows the species. The deformation, that is the impact a species has on the fitness landscape, could also be made dependent on additional external parameters. However, in our view, the model should be made as simple as possible.

3 Experimental Results

Experiments were performed on a two-dimensional fitness landscape. We have used circular boundary conditions. MPEG as well as AVI movies of these experiments are available for download from[1]. The source code for these experiments is also available for download.

The different parameter settings which were used are shown in Table 1. We used 40 species due to the fact that the size of the landscape has increased compared to the experiments on the one-dimensional landscape. A flat fitness landscape was used for experiments 1 through 7 while experiments 8 and 9 used a non-flat fitness landscape. For the latter two experiments, 100 Gaussian hills (with the same standard deviation that was used for the depression caused by the species) were distributed uniformly over the entire area.

Figure 3 shows the results for experiments 1, 3 and 5. For all three of these experiments, the species spread out evenly over the environment and then move over the environment. For experiment 2, we observe a clumped shift behavior with continued motion. Figure 4 shows the results. The number in the upper left hand corner shows the simulation time step. Eventually, the species synchronize their motion and all move in the same direction. We consider this to be an arms race type of movement as illustrated in Figure 2. The results of experiments 4, and 6 are shown in Figure 5 and 6 respectively. For both of these experiments, we observe a mixture between cyclic behavior and continued motion. This is best observed in the movie or by using the program code.

[1] `http://www.ra.cs.uni-tuebingen.de/mitarb/ebner/welcome.html`

Table 1. Parameter settings which were used for the experiments on the two-dimensional landscape

Exp.	Species	Update Rule	Latency	Hills	Observed Behavior	Fig.
1	40	Eqn (1)	0	0	spread out then shift	3
2	40	Eqn (1)	50	0	clumped shift (arms race)	4
3	40	Eqn (2)	0	0	spread out then tiny shift	3
4	40	Eqn (2)	50	0	cyclic	5
5	40	Eqn (3)	0	0	spread out then shift	3
6	40	Eqn (3)	2	0	cyclic	6
7	40	Eqn (3)	50	0	arms race	7 & 8
8	40	Eqn (3)	0	100	stasis	9
9	40	Eqn (3)	50	100	arms race	10

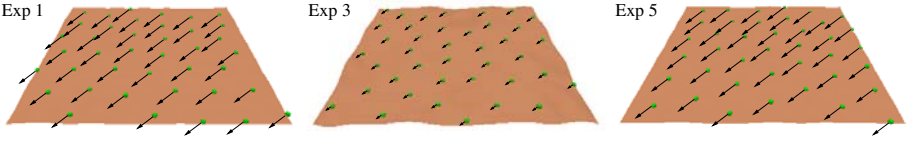


Fig. 3. Experiment 1, 3 and 5: synchronous shift of the species. The movement is very small for experiment 3.

A latency of 50 for experiment 7 resulted in an arms race similar to that of the one-dimensional case. The first few steps and the resulting arms race are shown in Figure 7. However, it also shows a very interesting difference. Even though we work with separate species in our model, converged points in the search space can be considered to be a single species. In the two-dimensional environment, the racing species do not all arrive at the same point as they climb out of the depression - rather they spread-out across the rim as can be seen in Figure 8. This slight stochastic separation is exaggerated by the competitive relationships of the species dynamics and occasionally produces a division - separate depressions each with a separate sub-set of species engaged in an arms race. This spontaneous speciation does not involve island models or other mechanisms of segregation such as reproductive isolation from one another [6] or mate preference due to marker traits [2]. In our model, speciation occurs through the coupled dynamics of the coevolving species. Thus, in our model sympatric

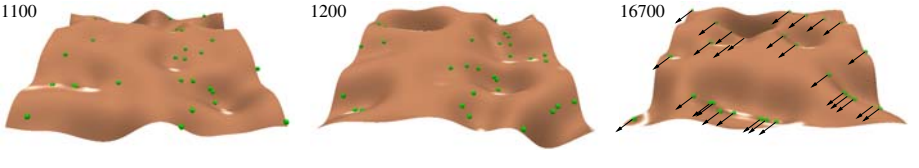


Fig. 4. Experiment 2: clumped shift behavior after convergence

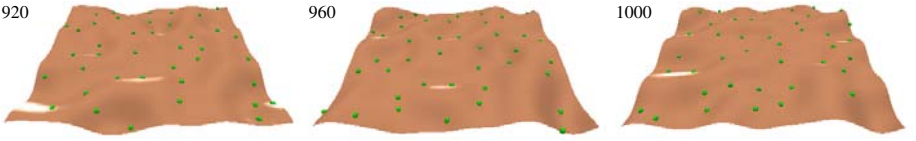


Fig. 5. Experiment 4: a mixture between cyclic behavior and continued motion

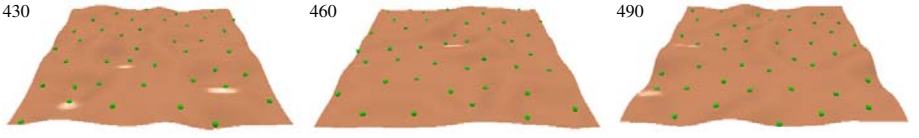


Fig. 6. Experiment 6: A latency of 2 produces cyclic behavior

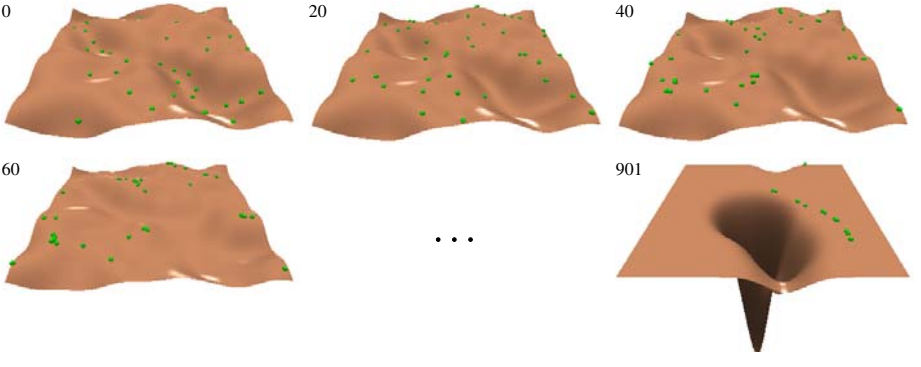


Fig. 7. Experiment 7: arms race between different species

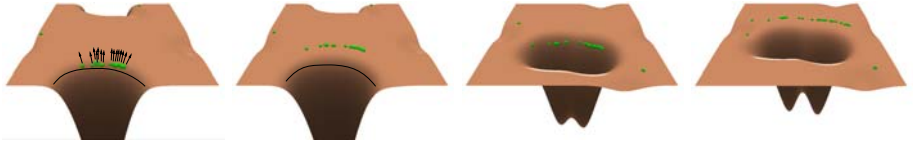


Fig. 8. Spontaneous speciation occurs during experiment 7. The species spread along the rim of the depression. This causes the creation of several sub-sets of species engaged in an arms race.

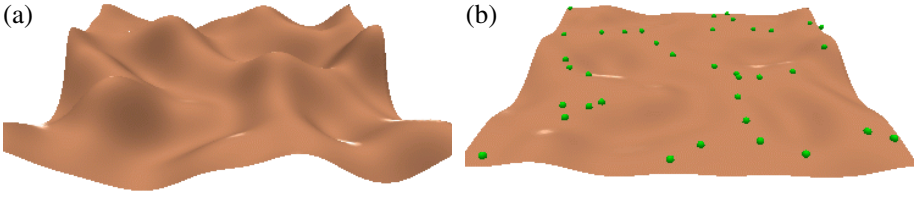


Fig. 9. (a) Non-flat environment (used for experiments 8 and 9). (b) Experiment 8: The species spread over the landscape to avoid the negative influence of other species and to exploit and fitness advantages present in the landscape.

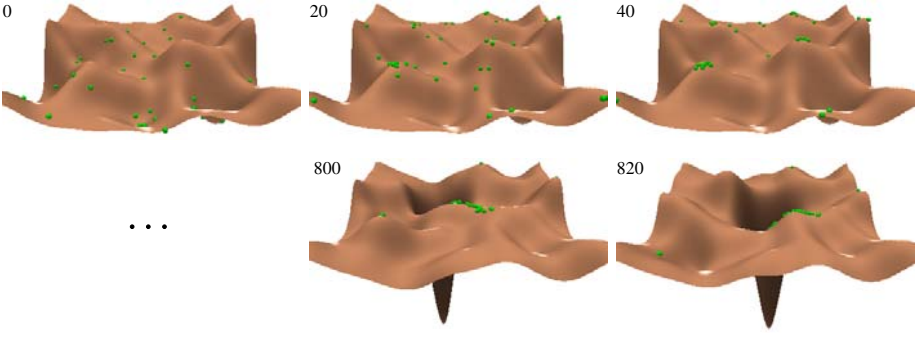


Fig. 10. Experiment 9: An arms race occurs. The species can eventually sweep over all local optima.

speciation [15], speciation without isolation, is also possible without modeling preferences for ecological traits or preferences for certain marker traits of other individuals.

For experiments 8 and 9 we have used the non-flat environment (shown in Figure 9(a)). Experiment 8, which used a latency of 0, resulted in a state of stasis with small local oscillations (shown in Figure 9(b)). This figure shows very clearly how competitive coevolution is able to automatically promote a niching behavior that distributes the species so as to equalize the landscape - the almost flat resulting landscape (as compared to the underlying landscape shown in Figure 9(a)) indicates that every part of the fitness landscape is being used near optimally. Experiment 9 with a latency of 50 resulted in an arms race. The result of this experiment is shown in 10.

We now address the question of why the use of a latency factor causes the species to explore the entire landscape. Essentially species adapt to the nearest local optimum. Consider species adapting to a non-flat fitness landscape with a latency factor of zero. The species immediately deform the landscape but each species still adapts towards a local optimum. The result is a stationary landscape where the species are distributed over the landscape in such a way that any move of the species would result in a lower fitness value for that species. Now suppose

that a sufficiently large latency factor is involved. If this is the case, then the species adapt towards the nearest optimum. However, due to the latency factor, the deformation caused by the species follows the species after some time. This local optimum invites other species to climb towards it as long as the combined negative influence of other species is smaller than the height of this optimum. Once several species have reached the same optimum and the combined negative influence of all of these species is larger than the height of this optimum, then the deformation pushes the species away from its previously optimal position in phenotype space. The species now have to climb towards new optima.

When several species climb towards the same optimum, they initially clump together before they are pushed away from that optimum. The result is a Red Queen type of arms race where the species are distributed along the rim of the depression. This arms race lasts until the species are sufficiently distant from each other. Note that the direction the species approach the optimum and also the exact timing of this approach influences the way the species are pushed away from the optimum. The result is that over time the species may reach any point of the landscape provided that in the model, the combined depression caused by the species is sufficiently large to push the species away from the highest optimum of the landscape.

4 Conclusion

We have developed a simple evolutionary model which is used to investigate the dynamics of competitive coevolution. Due to the simplicity of the model, it is easy to implement and easy to understand. Most importantly, it can be calculated fast enough which allows us to visualize the behavior in real time. Adaptation of a species to its environment is modelled as a local hill-climbing process which moves the population mean towards the nearest local optimum. Each species has an influence on the environment where it is located. In our model, each species deforms the fitness landscape and thereby interacts with other species located nearby. This allows us to study the dynamics of coevolving species. We have observed several qualitatively different behaviors. Depending on the choice of parameters, the species distribute evenly over phenotype space or engage in an arms race. In case of an arms race, all of phenotype space is explored. The determining factor which causes the switch between the two regimes was the latency parameter. The latency parameter determines the time lapse for a species to have an impact on its current environment. This model has shown that such delays in adaptive interactions can have a significant effect on the dynamics of coevolving species.

References

1. Dawkins, R., Krebs, J.R.: Arms races between and within species. Proc. R. Soc. Lond. B 205, 489–511 (1979)
2. Dieckmann, U., Doebeli, M.: On the origin of species by sympatric speciation. Nature 400, 354–357 (1999)

3. Ebner, M., Watson, R.A., Alexander, J.: Co-evolutionary dynamics on a deformable landscape. In: Proceedings of the 2000 Congress on Evolutionary Computation, San Diego, CA, vol. 2, pp. 1284–1291. IEEE Press, Los Alamitos (2000)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Reading (1989)
5. Grefenstette, J.J.: Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In: Proc. of the 1999 Congress on Evolutionary Computation, vol. 3, pp. 2031–2038. IEEE Press, Washington (1999)
6. Higgs, P.G., Derrida, B.: Genetic distance and species formation in evolving populations. Journal of Molecular Evolution 35, 454–465 (1992)
7. Kauffman, S.A.: The Origins of Order. Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
8. Lieberman, E., Hauert, C., Nowak, M.A.: Evolutionary dynamics on graphs. Nature 433, 312–316 (2005)
9. Morrison, R.W., Jong, K.A.D.: A test problem generator for non-stationary environments. In: Proc. of the 1999 Congress on Evolutionary Computation, vol. 3, pp. 2047–2053. IEEE Press, Washington (1999)
10. Nowak, M.A., Sigmund, K.: Evolutionary dynamics of biological games. Science 303, 793–799 (2004)
11. Ray, T.S.: Is it alive or is it GA? In: Belew, R.K., Booker, L.B. (eds.) Proc. of the 4th Int. Conf. on Genetic Algorithms, University of California, San Diego, pp. 527–534. Morgan Kaufmann Publishers, San Mateo (1991)
12. Spector, L., Klein, J.: Evolutionary dynamics discovered via visualization in the breve simulation environment. In: Workshop Proc. of the 8th Int. Conf. on the Simulation and Synthesis of Living Systems, Sydney, Australia, pp. 163–170. University of New South Wales (2002)
13. Stenseth, N.C., Smith, J.M.: Coevolution in ecosystems: Red queen evolution or stasis? Evolution 38(4), 870–880 (1984)
14. Taylor, C., Fudenberg, D., Sasaki, A., Nowak, M.A.: Evolutionary game dynamics in finite populations. Bulletin of Mathematical Biology 66, 1621–1644 (2004)
15. Tregenza, T., Butlin, R.K.: Speciation without isolation. Nature 400, 311–312 (1999)
16. Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Proc. of the 1999 Congress on Evolutionary Computation, vol. 3, pp. 1843–1850. IEEE Press, Washington (1999)
17. Valen, L.V.: A new evolutionary law. Evolutionary Theory 1, 1–30 (1973)
18. Wright, S.: The roles of mutation, inbreeding, crossbreeding and selection in evolution. In: Proc. of the 6th Int. Congress on Genetics, Ithaca, NY, pp. 356–366 (1932)

Evolving Individual Behavior in a Multi-agent Traffic Simulator

Ernesto Sanchez, Giovanni Squillero, and Alberto Tonda

Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy
{ernesto.sanchez,giovanni.squillero,alberto.tonda}@polito.it

Abstract. In this paper, we illustrate the use of evolutionary agents in a multi-agent system designed to describe the behavior of car drivers. Each agent has the selfish objective to reach its destination in the shortest time possible, and a preference in terms of paths to take, based on the presence of other agents and on the width of the roads. Those parameters are changed with an evolutionary strategy, to mimic the adaptation of a human driver to different traffic conditions. The system proposed is then tested by giving the agents the ability to perceive the presence of other agents in a given radius. Experimental results show that knowing the position of all the car drivers in the map leads the agents to obtain a better performance, thanks to the evolution of their behavior. Even the system as a whole gains some benefits from the evolution of the agents' individual choices.

Keywords: Multi-agent systems, Evolution, Traffic simulation.

1 Introduction

Road traffic congestion is a crucial problem, the short-range consequences of which can vary from delays to decreased throughput of vehicles. Long-range consequences include reduced safety, environmental pollution, and reduced economic competitiveness. This problem is becoming more intense, not only in western cities but also in countries where the presence of cars, once scarce, is growing at an alarming rate.

From websites displaying the current traffic conditions [1] to collections of traffic control strategies [2] available online, information technology is playing a vital role in the development of new approaches to traffic control, even by simply providing the means to evaluate innovative methodologies, by means of sensors, databases and data mining.

In this context, simulations are heavily employed to test the possible outcome of new strategies, and often multi-agent systems are chosen as a simulation tool. An agent is defined as a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks. A multi-agent system (MAS) is a collection of software agents that work in conjunction with each other. They may cooperate or they may compete, or some combination of the two, but there is some common infrastructure that result in the collection being a 'system', as opposed to simply being a disjoint set of autonomous agents [3]. Each agent in the MAS tries to achieve some individual or collective task.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 11–20, 2010.

Many works on MAS have been led around road traffic issues. For instance, agents can be used for traffic management. In [4], the authors propose an application based on coordination of agents to diagnostic and inform drivers about traffic problems in a located area. Traffic-related issues are also the basis for [5], [12] and [13]. In [8] a multi-agent simulation is used to validate three different self-organizing methods aimed at optimizing the configuration of traffic lights in a city.

In [6] Doniec et al. experimented traffic simulation by modeling an intersection between two real roads, and studying the behavior of agents representing car drivers who could choose whether to abide by the traffic rules or ignore them, with the selfish objective to minimize the time they had to wait in order to cross the intersection. They demonstrated that an accurately modeled MAS simulated the traffic trend effectively going on during the day in the real world intersection.

To enhance a generic MAS traffic simulator, we propose to populate it with agents able to evolve their behavior through an evolutionary strategy, mimicking human ability to adapt to changing traffic conditions. Behaviors are optimized to provide a more realistic simulation, that is, the result of the whole evolution process is a *single* realistic scenario. The simulator is not intended to analyze evolution. We demonstrate that the agents significantly improve their performance when given information on the position of other cars, thanks to the evolution in their preference on roads to go through. Since the change of behavior of an agent may trigger the change of behavior of other agents, this interaction creates a co-evolutionary system. However the work focuses on the global properties of the system, rather on an analysis of the co-evolution itself. In section 2, we illustrate the specifications for the model we built. Agents' behavior and evolution are discussed in section 3. Section 4 shows the results we obtained through experimental evaluation of a simple MAS. Section 5 contains the conclusions we were able to draw from the experience.

2 Model Specifications

In this work we use a simplified MAS to simulate the traffic in the central part of a typical medium-sized European town (e. g. with a population of 20,000 – 30,000 people), which usually has different kinds of roads, from the small alleys to the main street, from one-way roads to large two-way ones, with several roundabouts but no traffic lights. Each agent in the MAS models a car driver, and it has the objective of traveling from a starting point to an arrival point in the map.

Each agent in the MAS previously described possesses various properties:

a) a starting location;
b) an ending location;
c) agent's preferences when it has to choose a path.

Starting location and ending location are determined at the beginning of each run, for each agent: the first one is randomized between all the points in the map, while the second is chosen so that the path the car driver has to travel through is at least half the size of the map (both on the horizontal axis and on the vertical axis). This means that each agent has to pass through the center of the map in order to reach its destination,

thus increasing the probability of traffic jams. Each agent is started at a different time during the first seconds of the simulation.

The inclination of each agent to choose a certain road when it comes to a crossroad in the map is expressed by a series of weights associated to the agent itself. Those weights let an agent choose, for example, a longer but less crowded road over a shortest but very busy one, applying the formula that represents the hypothetical time needed to reach the next intersection on the road:

$$road\_weight = length\left(1 + \frac{w_1 agents\_num}{1 + w_2 width\_bool}\right)$$

where:

- *length* is the length of the road;
- *agents_num* is the number of agents on the road at the time when it is evaluated;
- *width_bool* is a Boolean value which express whether we are considering a two-lane or a one-lane road;
- *w1* and *w2* are the weights that lead the preference of the agent, making it choose wider roads over less crowded ones, or viceversa.

Every time an agents reaches a crossroad, it computes the value *road_weight* for each road departing from the crossroad, then for each road that road intersects, using *Dijkstra's algorithm* to find the path with minimum sum of *road_weights* based on the current situation. It is important to notice that this procedure is repeated each time an agent reaches a crossroad, because the parameter *agents_num* changes at each unit of time of the simulation as the agents are moving through the map. Thus, a car driver can change the path that it computed at the previous crossroad, as it perceives that the road he was going to take is now more busy, depending on the values associated to its weights.

Each agent has a perception of the situation of the roads around him up to a certain range, which can be set by the user of the simulation tool, expressed in the same units of length which are used for the map. For example, an agent could know how many agents are moving in all the roads in a radius of 100 m from its current position: in this situation, it would compute the *road_weight* for those roads as normal, but it could not obtain the parameter *agents_num* for streets outside that range, and thus would evaluate the remaining part of the map only on the basis of the *length* and *width* of the roads. It is important to notice that two agents may have the same starting and ending point in the map, and still follow two completely different paths, since their weights could make them choose two opposite ways from the beginning of the simulation.

The speed of an agent in the MAS is adjusted during each step of the simulation, on the basis of the number of cars that are currently traveling immediately ahead of him on the road the agent is going to take. If there are no cars, the agent will gradually increase its speed up to a maximum value, the city speed limit. If other cars are on its same path, vice versa, its speed will decrease by a value proportional to the number of cars and the width of the road, up to a complete stop.

3 Agents

The objective of each agent/car driver in the MAS is to travel from a starting point to an arrival point in the map. A driver is rewarded when he manages to minimize its own traveling time. It is important to notice that there is no common goal for all the agents: each one tries to reach its objective working independently from all the others. Despite this, we can have an index of how well the whole system is performing by measuring the average speed and the average time needed to reach the arrival point, considering all the agents roaming through the map.

3.1 Agents Structure

The genotype of each agent is a series of weights (*w1, w2*) that describe its behavior: those values represent the preferences of the agent when he is about to choose its way out of a crossroad, based on the number of other agents/cars on each street and on the number of lanes of the roads.

3.2 Evolution

In order to evaluate whether an optimal choice of path for a single agent could lead to an overall improvement in the mobility on all the roads, we chose to evolve the behavior of each agent, running the simulation through multiple generations.

There is, in fact, a series of problems that arise when we try to evaluate the global performance of this system, since each agent has a selfish objective and will not cooperate with other agents to reach it. Also, it is unlikely that the system itself will reach a stable state at a certain generation, because the change of behavior of an agent could potentially influence all the others: given that a car driver takes into account the presence of other cars on the roads it could choose, even a single agent modifying its path could lead to a chain-reaction of behavior modification for a great number of car drivers in the MAS. In order to avoid dramatic fluctuations of the paths and to reflect the behavior of human car drivers (that seldom change their track of choice), only a given percentage of the agents spawns a child each generation.

3.3 Agents Evolution

The evolution is similar to an *evolutionary strategy* (1+1): at each generation step, a single individual, which is represented by the weights associated to an agent, has a certain probability to produce a child. The child is then evaluated by making the agent behave on the basis of the new weights generated. It is important to notice that there is no such thing as a "population of all the agents": each agent stores a population made of a single individual that spawns a single child.

The genotype of each individual is a vector of real values (*w1, w2*): the new individual is created by mutation. A random value, obtained through a Gaussian distribution with mean 0, is added to each component of the vector. The selection is deterministic: the child is compared to the parent, and if its fitness value is better, it becomes the new parent; otherwise, it is discarded. The fitness, in our case, is based on the time it takes the agent to reach its destination: the lesser, the better. The population at each step is

thus composed by two individuals, parent and child. We made this choice because the fitness value can be properly compared only between individuals with the same starting and ending points on the map: since each agent represents a car driver, we are modeling the fact that a human in this situation would probably learn from the experience, changing the parameters on which he makes his choices on the basis of his past experiences.

Every agent's evolution is independent from the other agents, since car drivers, unluckily, do not cooperate with each other to minimize their travelling time. There is, however, a diffused co-evolution, since the behavior of each agent could influence the choices of all the others: one of the parameters taken into account when a car driver is selecting a path, is the number of cars on each road it could take. When even a single agent modifies its behavior, and consequentially its path, it obviously changes the number of cars that are going on a certain road: this could lead to a chain reaction where a great number of other agents would change path because of that increase/decrease in the quantity of cars on that road.

4 Experimental Evaluation

In order to experimentally evaluate our framework, we created a simplified model of a medium-sized downtown, to easily verify whether the results we would obtain were coherent to what we expected, while keeping the time needed for a simulation within reasonable parameters.

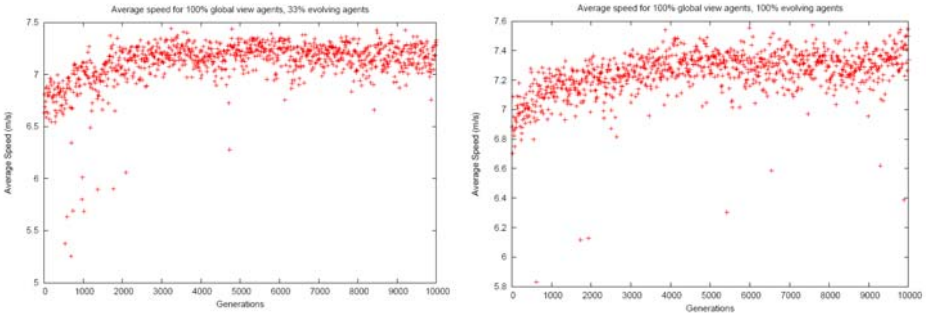
In our simplified model:

- all cars/agents are represented as points;
- each crossroad has up to four roads departing from it;
- crossroads are not regulated by traffic lights, a common occurrence in small to medium urban areas as the one in Figure 1, where roundabouts have replaced traffic lights;
- roads can only be vertical (crossing the map north to south) or horizontal (crossing the map east to west);
- the map of the city downtown is 0.16 Km^2, a square with an edge of 400 m, since we used meters as the unit of length to describe the roads and the cars, and comprehends 12 different roads, eight 400 m single lane roads, four 200 m single lane and two double lane 400 m ones.
- During each simulation, we chose to keep some parameters fixed, in order to easily compare the experimental results:
- the number of agents on the map at the same time is 400, which makes our city downtown quite crowded. Such a density is chosen to increase evolutionary pressure;
- at the beginning of the simulation, 10 agents are placed on their starting point every second. Each vehicle in a simulation run has the same initial and final location in every generation;
- once every 10 generations, we chose to run a simulation where no agent produced offspring, in order to check what is the best strategy, expressed as a set of weights, obtained for each agent up to that point;
- each car driver/agent can reach a maximum speed of 13 m/s, which is about 46.8 km/h or 29.1 MPH, slightly under the speed limits that are enforced in most

downtown streets. Also, when the traffic is heavy, it is very unlikely that a driver could run at a higher speed;
- when any agent reaches a crossroad and finds that its average speed computed on the last 10 seconds of simulation is under 7 m/s (about 25.2 km/h or 15.7 MPH), it will compute again the path from its current location to the destination, using an updated value for the number of agents on each road;
- each agent has a complete knowledge of the streets' map;
- depending on the simulation, a certain number of agents possesses a knowledge of the position of other agents the whole map, while the rest has a *local view*, which means that it can perceive the presence of cars only in a radius of 150 m;
- we consider that each car occupies 6 m in a certain road, because most cars are 5 m long, and when they are in a line it is wise to keep a distance of at least 1 m from the other cars. This parameter is used by the agents to detect the presence of other car drivers inside their sight radius.

4.1 Complete Knowledge

In the first set of runs, we tested a simulation where every agent had a complete knowledge of the position of all other agents. Only a third (33%) of the agents, selected randomly, would produce an offspring at each generation step. We let our system evolve for 10,000 generations, and then we evaluated the average speed (considering all cars in the MAS) each 10 generations, thus plotting only the simulations with all the best individuals.



(A) **(B)**

Fig. 1. On the X axis, number of generations. On the y axis, average speed (m/s).
(A) Average speed for 100% global view agents, 33% evolving agents.
(B) Average speed for 100% global view agents, 100% evolving agents.

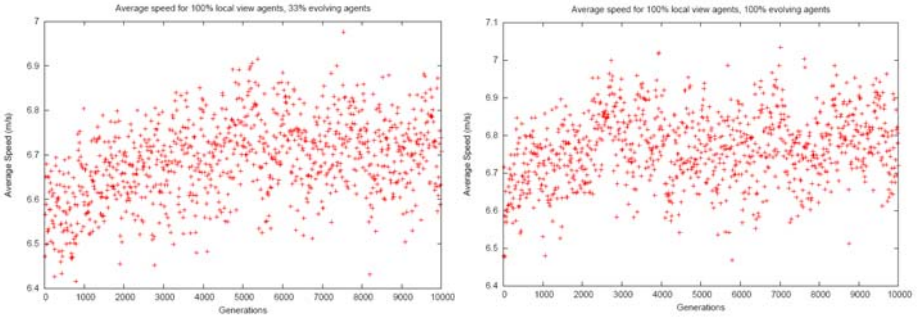
We can see that, while there is a fluctuation in the average speed (due to the agents changing their paths), it quickly increases from 6.5 m/s to over 7 m/s during the first 3,000th generations, then it keeps shifting between 7.25 and 7.3 m/s, with a peak just under 7.5 m/s. There are some generations, in the beginning of the simulation, where the average speed drops to values around 5.5 m/s, probably because the system is adjusting to the new paths chosen by a significant number of agents. After 5,000 generations, however, the fluctuations become smaller and smaller.

Increasing the number of agents that generate an offspring at each step could improve the performance of the system: thus, we ran another series of simulations where we raised the percentage of car drivers trying new preferences at each generation up to 100%.

As we can see the average speed rises faster during the first 3,000 steps, while the fluctuations in the later generations seem to be a little stronger than in the previous run, with drops in speed even after 5,000 generations.

4.2 Partial Knowledge

In a second time, we tested a set of simulations where no agent had a complete knowledge of the position of the other agents: thus, 100% of the car drivers on the map could perceive other cars only in a radius of 150 m from their current position. In this first simulation, only 33% of the agents change their preferences at each generation.



(A) **(B)**

Fig. 2 . On the X axis, number of generations. On the y axis, average speed (m/s).
(A) Average speed for 100% local view agents, 33% evolving agents.
(B) Average speed for 100% local view agents, 100% evolving agents.

As we expected, even from the first generation the average speed is lower than the values obtained from the previous experiments, where all the car drivers could perceive the presence of every car driver on every road. The average speed rises far slowly and, even if there are generations with peaks just below 7 m/s, the values are centered around 6.7 m/s. The improvement in the average speed is not so distinct as in the previous runs, and even in the last generations we have points that are at the same level of the initial value.

Increasing the number of car drivers trying new preferences at each generation proved useful when we had agents with global information on the position of other cars: running a set of simulations with 100% of partial-knowledge agents and 100% of evolving ones produced the following graphic.

While there are various peaks over 7 m/s, the shape of the graphic is very similar to the previous simulation where all the agents had only partial information.

Table 1. Average time for an agent to reach its destination, under different conditions

Type of experiment		**Average on the first 5 generations**	**Average on the last 5 generations**
100% complete knowledge	33% evolving	61.50 s	56.30 s
	100% evolving	60.09 s	55.10 s
100% partial knowledge	33% evolving	63.29 s	61.34 s
	100% evolving	63.73 s	61.33 s

The information in Table 1 shows that the average time the agents need to reach their destination drops in a significant way during the simulations where all the agents had all the global information on the map, while the improvement in the experiments where the agents had only information on the position of the other cars in a limited radius is much smaller.

From the experimental results, we can see that information about the position of other cars in a heavy traffic situation like the one we simulated is surely useful, as we expected, both for the single agent and for the system. Even if each agent used the information it had in order to selfishly maximize his speed, without any cooperation with the other car drivers, this behavior proved functional to the whole system: the average speed of all the car drivers increased from the first generation up to a certain value, and kept fluctuating around that speed.

We can notice that the increment is present in every simulation, even those where all the agents can perceive the presence of other cars only in a radius of 150 m: we can conclude that even partial information, provided in real-time to car drivers able to evolve their behavior like humans do, could help lightning the traffic in the downtown of great cities.

4.3 Mixed Agents

By setting up our MAS with both local-view agents and global-view agents, we expected to find a slower increase in the average speed (thus, a slower decrease of the average time the agents need to arrive to their destination) than in the "complete knowledge" case, while on the contrary obtaining better results than in the "partial knowledge" simulation. We also assumed that, even if the presence of global-view agents would surely improve the average performance of the system, above a certain percentage of global-view agents there would be no further improvements. Thus, we ran several experiments with mixed agents, in various proportions, while keeping the other parameters fixed. The percentage of evolving agents is always 100%, and the local view radius is 150 m.

Our results confirm what we supposed: the average time the agents need to reach their arrival location drops steadily as the percentage of local-view agents drops and the percentage of global-view agents increases. This could mean that in a real situation, even if not all the drivers could have access to all the information, there could be

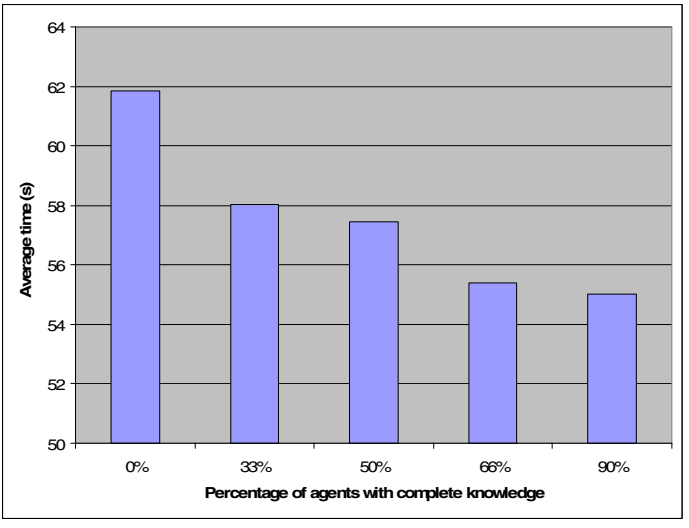


Fig. 3. Different arrival times with an increasing percentage of complete knowledge information agents in the population

an improvement in the traffic given as long as a certain percentage of car drivers have complete knowledge of the position of other cars.

5 Conclusions

We presented a MAS populated by agents able to evolve their behavior. Each agent represents a car driver starting from a random point in the map, with the objective to reach another random point, with the selfish objective of minimizing its traveling time. Every agent is able to evolve its choices through a series of generations, and it has a certain knowledge of the whole map. We tested the system with a number of agents chosen in order to model a situation where traffic is heavy, for example at the start or at the end of a business day, and providing each agent with information that could help a real-world driver, able to change its behavior. Our experimental results show that, even if there is no cooperation between agents, the average time used to arrive at their goal will gradually decrease generation after generation, finally starting to dynamically shift around a minimum value. This process is faster and the minimum reached is lower when every agent has complete information of its surroundings and of the position of every other agent. We can conclude that data obtained is consistent with the expected results, and thus the evolution of agents' behavior could provide more realistic data in traffic simulations. Future works will include large scale simulations with evolvable agents, implemented with open-source microscopic traffic simulation packages [10][11].

Acknowledgements

Our thanks to Chiara Patané for developing the software used in the simulations.

References

1. http://trafficinfo.lacity.org/index.html
2. Ding, W., Marchionini, G.: A Study on Video Browsing Strategies. Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD (1997)
3. http://www.agtivity.com/def/multi_agent_system.htm
4. Hernandez, J.Z., Ossowski, S., Garca-Serrano, A.: Multiagent architectures for intelligent traffic management systems. Transportation Research Part C: Emerging Technologies, 10, 473–506 (2002)
5. Bazzan, A.L.C.: A distributed approach for coordination of traffic signal agents. Journal of Autonomous Agents and Multi-Agent Systems (10), 131–164 (2005)
6. Doniec, A., Espie, S., Mandiau, R., Piechowiak, S.: Non-normative behaviour in multi-agent system: some experiments in traffic simulation. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2006, December 18-22, pp. 30–36 (2006)
7. Balan, G., Sean, L.: History-based Traffic Control. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 616–621 (2006)
8. Gershenson, C.: Self- Organizing Traffic Lights. Centrum Leo Apostel (2005)
9. Maroto, J., Delso, E., Félez, J., Cabanellas, J.M.: Real-time traffic simulation with a microscopic model. IEEE Trans. Intell. Transp. Syst. 7, 513 (2006)
10. Krajzewicz, D., Bonert, M., Wagner, P.: RoboCup 2006 Infrastructure Simulation Competition (2006)
11. Barceló, J., Codina, E., Casas, J., Ferrer, J.L., García, D.: Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. Journal of Intelligent and Robotic Systems 41(2-3) (January 2005)
12. Kesting, A., Treiber, M., Helbing, D.: Agents for Traffic Simulation, Contribution to Agents, Simulation and Applications. In: Uhrmacher, A., Weyns, D. (eds.) eprint arXiv:0805.0300 (May 2008)
13. Yang, J., Deng, W., Wang, J., Li, Q., Wang, Z.: Modeling pedestrians' road crossing behavior in traffic system micro-simulation in China. Transportation Research Part A: Policy and Practice 40(3), 280–290 (2006)

On Modeling and Evolutionary Optimization of Nonlinearly Coupled Pedestrian Interactions

Pradyumn Kumar Shukla

Institute of Numerical Mathematics, TU Dresden, 01062 Dresden, Germany
Institute AIFB, Karlsruhe Institute of Technology, 76133 Karlsruhe, Germany
pradyumn.shukla@kit.edu

Abstract. Social force based modeling of pedestrians is an advanced microscopic approach for simulating the dynamics of pedestrian motion. The developments presented in this paper extend the widespread social force model to include improved velocity-dependent interaction forces. This modeling considers interactions of pedestrians with both static and dynamic obstacles, which can be also be effectively used to model pedestrian-vehicle interactions. The superiority of the proposed model is shown by comparing it with existing ones considering several thought experiments. Moreover, we apply an evolutionary algorithm to solve the model calibration problem, considering two real-world instances. The objective function for this problem comes from a set of highly nonlinear coupled differential equations. An interesting feature that came out is that the solutions are multi-modal. This makes this problem an excellent example for evolutionary algorithms and other such population based heuristics algorithms.

Keywords: microscopic modeling, complex systems, optimization.

1 Introduction

In the past, there have been many approaches to modeling of pedestrians (see [9,8] and references therein). The social force model [8,7], for example, is a widely applied approach, which can model many observed collective behaviors. In this contribution, we will discuss existing variants of the social force model and propose a more realistic treatment of avoidance maneuvers. As the concept of safety is directly related to the velocity of the pedestrians, we will formulate a velocity-dependent social force model. The new model is shown to predict realistic microscopic pedestrian-pedestrian and pedestrian-static obstacle avoidance maneuvers. The new model can also model pedestrian-dynamics obstacle interactions and hence can also be used to model pedestrian-vehicle interactions, a topic which has been neglected in the past.

Calibration of any pedestrian model is an important area of research as only then the model could be applied for real-world simulations [10]. However, we have found limited studies on calibration of pedestrian models. For the model developed in this paper, we apply an evolutionary algorithm to calibrate and

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 21–30, 2010.

find an *optimal set of parameters*. Many factors play a role in calibration, like cultural differences, age-factor, fraction of handicapped persons among others. We take some macroscopic properties from previous real-world scenarios, like the distance headway vs. speed and flow-density relationships [1]. These studies also consider cultural-differences in them. We then apply an evolutionary algorithm to obtain an optimal parameter set for the two real-world situations. An interesting feature that came out from the optimization is that the solutions are multi-modal. Moreover, the objective function for this problem comes from a set of highly nonlinear coupled differential equations. We believe, that stochasticity of the objective function together with multi-modality of solutions makes this problem an excellent candidate for evolutionary algorithm.

The paper is organized as follows: Section 2 introduces different variants of the existing social force and similar models while in Section 3 we discuss limitations of the existing modeling approaches. Improved velocity-dependent forces are also introduced in the same section and the performance of all the models on different thought experiments are presented. Section 4 presents the evolutionary algorithm which is used to find an optimal set of parameters for the new model. Concluding remarks are made in the last section.

2 Existing Models

The social force model can be underpinned with a social science model of behavioral changes proposed by Lewin [11]. He has introduced the idea that behavioral changes were guided by so-called *social fields* or *social forces*. This idea has been put into mathematical form by Helbing [4] and applied to pedestrian motion, and vehicular traffic [14]. The social force model for pedestrians assumes that each individual α is trying to move in a desired direction $\boldsymbol{e}_\alpha$ with a desired speed v_α^0, and that it adapts the actual velocity $\boldsymbol{v}_\alpha$ to the desired one, $\boldsymbol{v}_\alpha^0 = v_\alpha^0 \boldsymbol{e}_\alpha$ within a certain relaxation time τ_α. The velocity $v_\alpha(t) = d\boldsymbol{r}_\alpha/dt$, i.e. the temporal change of the location $r_\alpha(t)$, is itself assumed to change according to the acceleration equation

$$\frac{d\boldsymbol{v}_\alpha(t)}{dt} = \boldsymbol{f}_\alpha(t) + \boldsymbol{\xi}_\alpha(t) \tag{1}$$

where $\boldsymbol{\xi}_\alpha(t)$ is a fluctuation term and $\boldsymbol{f}_\alpha(t)$ the systematic part of the acceleration force of pedestrian α given by

$$\boldsymbol{f}_\alpha(t) = \frac{1}{\tau_\alpha}(v_\alpha^0 \boldsymbol{e}_\alpha - \boldsymbol{v}_\alpha) + \sum_{\beta(\neq\alpha)} \boldsymbol{f}_{\alpha\beta}(t) + \sum_i \boldsymbol{f}_{\alpha i}(t)\,. \tag{2}$$

The terms $\boldsymbol{f}_{\alpha\beta}(t)$, $\boldsymbol{f}_{\alpha i}(t)$ denote the repulsive forces describing the attempts to keep a certain safety distance to other pedestrians β and obstacles i. The fluctuations term $\boldsymbol{\xi}_\alpha(t)$ reflects random behavioral variations arising from deliberate or accidental deviations from optimal strategy of motion. The above equation are nonlinearly coupled Langevin equations and can be solved numerically using Euler's method. In very crowded situations (not considered in this paper), additional physical contact forces come into play [7].

There are at least six formulations of the repulsive pedestrian interaction forces $\boldsymbol{f}_{\alpha\beta}(t)$ in the literature of which five are based on social force model and one based on magnetic force model.

1. **Circular formulation:** In Ref. [7], the psychologically induced repulsive force of pedestrian β on α was, for reasons of simplicity, assumed to depend only on the distance $d_{\alpha\beta} = \|\boldsymbol{r}_\alpha - \boldsymbol{r}_\beta\|$ between the centers of mass of two pedestrians. Moreover, it was assumed that

$$\boldsymbol{f}_{\alpha\beta}(t) = A\,\mathrm{e}^{(d-d_{\alpha\beta})/B}\frac{\boldsymbol{d}_{\alpha\beta}}{d_{\alpha\beta}} \tag{3}$$

   The model parameters A and B denote the interaction strength and interaction range, respectively, while $d \approx 0.6$m is the sum of radii of both pedestrians (i.e. the "pedestrian diameter"). $\boldsymbol{d}_{\alpha\beta} = (\boldsymbol{r}_\alpha - \boldsymbol{r}_\beta)$ represents the vector pointing from pedestrian β to α.
2. **Elliptical formulation I:** In Ref. [8], a first velocity-dependent interaction force was formulated. It was assumed that the repulsive potential

$$V_{\alpha\beta}(b) = V^0_{\alpha\beta}\,\mathrm{e}^{-b/\sigma} \tag{4}$$

   is an exponentially decreasing function of b with equipotential lines having the form of an ellipse that is directed into the direction of motion as shown in Fig. 1(left). The semi-minor axis $b_{\alpha\beta}$ was determined by

$$2b_{\alpha\beta} = \sqrt{(\|\boldsymbol{d}_{\alpha\beta}\| + \|\boldsymbol{d}_{\alpha\beta} - v_\beta\,\Delta t\,\boldsymbol{e}_\beta\|)^2 - (v_\beta\,\Delta t)^2} \tag{5}$$

   in order to take into account the length $v_\beta\,\Delta t$ of the stride (step width) of pedestrian β (where $v_\alpha = \|\boldsymbol{v}_\alpha\|$). The reason is that a pedestrian requires space for movement, which is taken into consideration by other pedestrians. Note that the $b_{\alpha\beta}$-value (hence the repulsive force) is same along the equipotential lines. The repulsive force is related to the repulsive potential as

$$\boldsymbol{f}_{\alpha\beta}(\boldsymbol{d}_{\alpha\beta}) = -\boldsymbol{\nabla}_{\boldsymbol{d}_{\alpha\beta}} V_{\alpha\beta}(b_{\alpha\beta}) = -\frac{dV_{\alpha\beta}(b_{\alpha\beta})}{db_{\alpha\beta}}\boldsymbol{\nabla}_{\boldsymbol{d}_{\alpha\beta}} b_{\alpha\beta}(\boldsymbol{d}_{\alpha\beta})\,. \tag{6}$$

3. **Elliptical specification II:** Recently, a variant of this approach has been proposed [13], assuming

$$2b_{\alpha\beta} := \sqrt{(\|\boldsymbol{d}_{\alpha\beta}\| + \|\boldsymbol{d}_{\alpha\beta} - (\boldsymbol{v}_\beta - \boldsymbol{v}_\alpha)\Delta t\|)^2 - [(\boldsymbol{v}_\beta - \boldsymbol{v}_\alpha)\Delta t]^2}\,. \tag{7}$$

   The special feature of this approach is its symmetrical treatment of both pedestrians α and β.
4. **Elongated formulation:** A further improved, elongated form of the repulsive interaction potential was proposed in Ref. [5], see Fig. 1(right). It was defined by the formula

$$V_{\alpha\beta}(b_{\alpha\beta}) = A\,\mathrm{e}^{-\sqrt{(\boldsymbol{d}_{\alpha\beta}\cdot\boldsymbol{e}_1)^2 + (\boldsymbol{d}_{\alpha\beta}\cdot\boldsymbol{e}_2)^2/(\gamma_\alpha)^2}/B}\,, \tag{8}$$

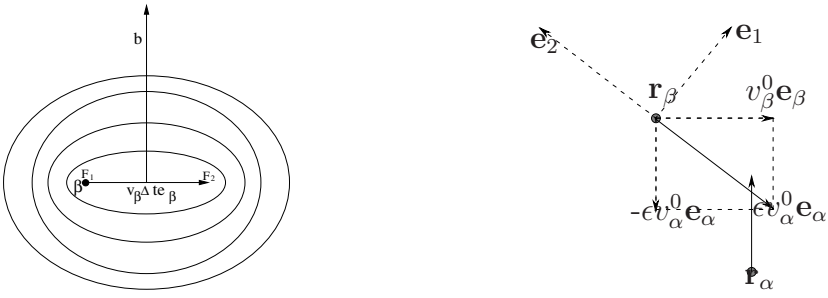


Fig. 1. Elliptical specification I (left) and Elongated specification (right) of pedestrian interaction forces

where A and B are model parameters and the variable $\gamma_\alpha(t)$ is given by

$$\gamma_\alpha = \begin{cases} \vartheta & \text{if } \boldsymbol{e}_2 \cdot \boldsymbol{d}_{\alpha\beta} \geq 0 \\ 1 + v_\alpha \, \Delta t & \text{otherwise.} \end{cases} \tag{9}$$

With

$$\boldsymbol{e}_2(t) := \frac{\epsilon v_\alpha^0 \mathbf{e}_{\alpha(t)} - v_\beta^0 \mathbf{e}_{\beta(t)}}{\epsilon v_\alpha^0 \mathbf{e}_{\alpha(t)} - v_\beta^0 \mathbf{e}_{\beta(t)}} \tag{10}$$

this takes into account not only a dependence on the length $v_\alpha \, \Delta t$ of pedestrian α's stride but also on the relative direction of motion. $\boldsymbol{e}_1(t)$ denotes a vector perpendicular to $\boldsymbol{e}_2(t)$.

5. **Centrifugal force formulation:** This formulation was proposed in Ref. [15] and the repulsive force as per this formulation is given by

$$\boldsymbol{f}_{\alpha\beta}(\boldsymbol{d}_{\alpha\beta}) := m_\alpha \frac{\{(\boldsymbol{v}_\beta - \boldsymbol{v}_\alpha) \cdot \boldsymbol{e}_{\alpha\beta} + \|(\boldsymbol{v}_\beta - \boldsymbol{v}_\alpha) \cdot \boldsymbol{e}_{\alpha\beta}\|\}^2}{2\|\boldsymbol{d}_{\alpha\beta}\|} \boldsymbol{d}_{\alpha\beta}. \tag{11}$$

where m_α is the mass of pedestrian α and $\boldsymbol{e}_{\alpha\beta}$ denotes the unit vector in the direction $\boldsymbol{d}_{\alpha\beta}$. This formulation improved the previous models in that it considers not only the relative velocity between pedestrian α and β but also the headway between them.

6. **Magnetic force formulation:** This formulation was proposed by Okazaki as early as in 1979. The repulsive force as per this formulation is sum of two forces one depends upon the headway while the other depends upon velocity of pedestrian α and direction of relative velocity. For the details we refer the reader to Ref. [12].

3 Limitations of Previous Approaches and a New Microscopic Model

In order to study the limitations of above formulations, let us consider four set of thought experiments.

Pedestrian-pedestrian head-on interactions. Let us start with a first experiment illustrated in Fig. 2. In this set of experiments we assume that two pedestrians α and β are walking close-by. In Case 1 and Case 2, Pedestrian α is moving towards pedestrian β with velocity $\boldsymbol{v_\alpha}^1$ and $\boldsymbol{v_\alpha}^2$ respectively. Assume that $\boldsymbol{v_\alpha}^1 > \boldsymbol{v_\alpha}^2$. One can easily see that the repulsive force exerted by β should be higher in Case 1. However, the Circular and Elliptical I formulation of the pedestrian interaction force do not take into account this velocity dependence and thus imply the same (unrealistic) social force in both situations. In Case 3, (when pedestrian β also moves towards α) consider the repulsive interaction force on β due to α. Now, the Circular and Elongated formulation of the pedestrian interaction force does not depend upon $\boldsymbol{v}_\alpha$ which is unrealistic. β perceives greater threat if $\boldsymbol{v}_\alpha$ is more and hence a greater repulsive force should result. In Case 4, (when pedestrian β moves away from α) consider the repulsive interaction force on α due to β. let us assume that the relative velocity is the same. Then a larger headway should result as effect of forces when speeds of pedestrians are more. This is because pedestrians can move closer when their speeds are small compared to when say, running. Now, the Elongated II formulation of the pedestrian interaction force only depends upon the relative velocity which is unrealistic. Thus from these set of experiments we conclude that, a realistic model should have velocity dependence of both interacting pedestrians.

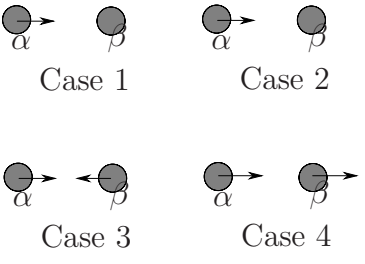


Fig. 2. Pedestrian-pedestrian head on interactions

Pedestrian-pedestrian sideways interactions. Let us start with a first experiment illustrated in Fig. 3. In this set of experiments we assume that two pedestrians α and β are walking sideways. In Case 1 they are on a collision course compared to Case 2. Let the angle φ between direction of motion of pedestrian α and the direction of the other pedestrian β be constant in both cases, i.e. in both cases anisotropy (usually a function of ψ) is constant. For the Elliptical I formulation of the interaction force, it can now be easily seen that the $b_{\alpha\beta}$-value in case A is smaller than the $b_{\alpha\beta}$-value in case B. Therefore, the repulsion force acting on pedestrian α should be higher in case A than in case B. This is consistent with expectations according to safety considerations. Similarly, the elongated formulation realistically predicts a greater interaction force in case A. In contrast to expectation, however, the Circular and Magnetic formulation predicts the same force in case A and

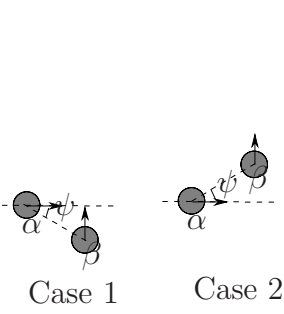


Fig. 3. Pedestrian-pedestrian sideways interactions

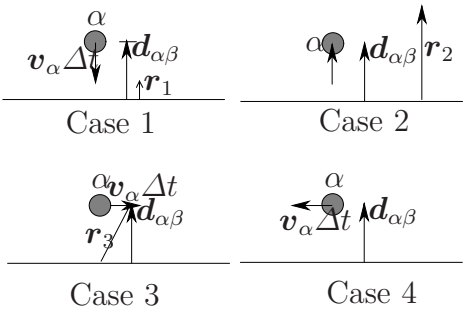


Fig. 4. Pedestrian-static obstacle interactions

case B, as the value of φ is the same. A realistic model should have also a function of relative velocity of interacting pedestrians.

Pedestrian-static obstacle interactions. Consider a pedestrian walking towards a wall with a given speed v_α, but four different orientations $\boldsymbol{e}_\alpha^k$ (with $k \in \{1, 2, 3, 4\}$), as shown in Fig. 4. This scenario can be treated analogously to the movement relative to a standing pedestrian β, which implies $\boldsymbol{v}_\beta = \boldsymbol{0}$. Then, as per Elliptical formulation II the repulsive force is a monotonously decreasing function of $b_{\alpha\beta}$ given by

$$2b_{\alpha\beta} = \sqrt{(\|\boldsymbol{d}_{\alpha\beta}\| + \|\boldsymbol{d}_{\alpha\beta} + v_\alpha \Delta t \boldsymbol{e}_\alpha\|)^2 - (v_\alpha \Delta t)^2} \tag{12}$$

according to Eq. (7). For all four cases, the values of $d := \|\boldsymbol{d}_{\alpha\beta}\|$ and $v_\alpha \Delta t$ (the step size of pedestrian α) are the same, but the values of $d_\alpha^k := \|\boldsymbol{d}_{\alpha\beta} + v_\alpha \Delta t \boldsymbol{e}_\alpha^k\|$ are different. We have $d_\alpha^1 < d_\alpha^3 = d_\alpha^4 < d_\alpha^2$, so that we find $F_\alpha^1 > F_\alpha^3 = F_\alpha^4 > F_\alpha^2$ for the magnitudes of the repulsive forces triggered by the wall (as the direction of the forces is perpendicular to the wall in all four cases.) This agrees well with experience, i.e. the anisotropic and orientation behavior of pedestrians are realistically reproduced by the elliptical force specification II. In contrast, the Elliptical model I implies $b_{\alpha\beta} = d_{\alpha\beta}$ according to Eq.5 and predicts the same force in all four cases, as does the circular model.

Pedestrian-dynamic obstacle interactions. Although they have never been thoroughly studied, they can treated in a manner similar to pedestrian-pedestrian interactions.

Based on the above description of the models it can be concluded that in order to model realistically the microscopic pedestrian interactions a realistic model should take into account (i) dependence on pedestrian's own velocity, (ii) relative velocity of interacting pedestrians, (iii) headway dependence. The well studied car-following theories in vehicular traffic also take into account all these three variables [6] while *none* of the above pedestrian interaction models takes into consideration all these variables. Based on the above discussion we propose

the following general repulsive force for pedestrian-pedestrian, static-obstacle, dynamic-obstacle microscopic interactions.

New Elliptical specification: In this we assume that

$$2b_{\alpha\beta} := \sqrt{\{(\|\boldsymbol{d}_{\alpha\beta}\| + \|\boldsymbol{d}_{\alpha\beta} - (\boldsymbol{v}_\beta - \boldsymbol{v}_\alpha)\Delta t\|)^2 - [(\boldsymbol{v}_\beta - \boldsymbol{v}_\alpha)\Delta t]^2\}/(1 + v_\alpha \Delta t))}\,.$$

With this, the repulsive force is given by

$$\boldsymbol{f}_{\alpha\beta}(\boldsymbol{d}_{\alpha\beta}, \boldsymbol{v}_\alpha, \boldsymbol{v}_{\alpha\beta}) = A\mathrm{e}^{-b_{\alpha\beta}/B} \cdot \frac{\boldsymbol{d}_{\alpha\beta}}{\|\boldsymbol{d}_{\alpha\beta}\|}. \tag{13}$$

Note that this model is a function of $\boldsymbol{d}_{\alpha\beta}$,$\boldsymbol{v}_\alpha$ and $\boldsymbol{v}_{\alpha\beta}$ and thus it can also be effectively used to model pedestrian interactions with dynamic obstacles like vehicles for example. For pedestrian-vehicle interactions (the same) Equation 13 is used, with the corresponding parameters A_o and B_o instead of A and B. Thus, for calibration the parameters A, A_o and B, B_o (and an anisotropy factor described shorty) need to be properly calibrated. The performance of all the models in summarized in Table 1. We can see from the table that the New Elliptical model is the *only* model that can successfully model all the avoidance maneuvers without any extra parameters. Note that pedestrians react more strongly to things happening in *front* of them than *behind*. This is termed as *anisotropy* and can be taken into account in any of the above discussed model by multiplying the force terms with the factor

$$\omega\left(\psi_{\alpha\beta}(t)\right) := \lambda_\alpha + (1 - \lambda_\alpha)\frac{1 + \cos(\psi_{\alpha\beta})}{2},$$

where $\psi_{\alpha\beta}$ is the angle between the vectors $\boldsymbol{d}_{\alpha\beta}$ and $\boldsymbol{e}_\alpha$ and λ_α is a constant.

Table 1. Variables considered in all the models are summarized. The models are evaluated for their performance on thought experiments.

Model	Variables considered	Unrealistic performance cases
Circular	$\boldsymbol{d}_{\alpha\beta}$	All
Elliptical I	$\boldsymbol{d}_{\alpha\beta}$, $\boldsymbol{v}_\beta$	Ped.-ped. head-on
Elliptical II	$\boldsymbol{d}_{\alpha\beta}$, $\boldsymbol{v}_{\alpha\beta}$	Ped.-ped. head-on
Elongated	$\boldsymbol{d}_{\alpha\beta}$, $\boldsymbol{v}_\alpha$	Ped.-ped. head-on
Centrifugal force	$\boldsymbol{d}_{\alpha\beta}$, $\boldsymbol{v}_{\alpha\beta}$	Ped.-ped. head-on
Magnetic force	$\boldsymbol{d}_{\alpha\beta}$, direction of $\boldsymbol{v}_{\alpha\beta}$	Ped.-ped. sideways
New Elliptical	$\boldsymbol{d}_{\alpha\beta}$, $\boldsymbol{v}_{\alpha\beta}$ *and* $\boldsymbol{v}_\alpha$	*None*

4 Evolutionary Parameter Optimization of the New Model

In this section we use an evolutionary algorithm to calibrate the New Elliptical model. In the next we describe the problem, algorithm and the results in detail.

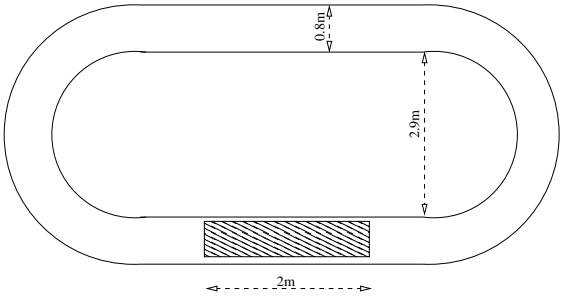


Fig. 5. Sketch of the simulation set-up

4.1 The Problem

We use an evolutionary algorithm to calibrate New Elliptical model in two different situations. These situations have been used and described in [1]. It is a corridor adopted from data collection in Germany and India. The situation is illustrated in Fig. 5. The length of the corridor is $17.3m$ and we are interested in the shaded rectangular region of length $2m$. With a number of pedestrians $N = 30$, the following speed-headway relation have been obtained the pedestrian data:

$$h = 0.22 + 0.89v \quad \text{(Indian conditions)} \tag{14}$$

$$h = 0.36 + 1.04v \quad \text{(German conditions)}, \tag{15}$$

where h, v denote the headway(m) and speed(m/s), respectively. We use this macroscopic relationship to calibrate our model. We simulate our model with $N = 30$ pedestrians in the corridor shown in Fig. 5. Data about headway and speed is collected pedestrians for the shaded region. The average of the absolute values between difference of observed and predicted (from the above $h - v$ equations) is used as the objective function. Obviously, this objective function

Table 2. Three set of optimal values of parameters obtained from the evolutionary algorithms. The values are given for both Indian and German conditions.

	A	B	A_o	B_o	λ	Objective function
Indian conditions	1.12	0.25	1.12	0.25	0.64	0.41
German conditions	1.96	0.32	1.12	0.25	0.92	0.42
	A	B	A_o	B_o	λ	Objective function
Indian conditions	2.04	0.23	1.12	0.25	0.46	0.41
German conditions	2.14	0.32	1.12	0.25	0.73	0.40
	A	B	A_o	B_o	λ	Objective function
Indian conditions	3.0	0.23	1.12	0.25	0.45	0.41
German conditions	3.6	0.53	1.12	0.25	0.67	0.42

needs to be minimized and, due to the difference between experimental data and simulation results, we would not expect this value to be zero.

4.2 The Algorithm

In our calibration method, we used a real coded evolutionary algorithm see [2]. The maximum generation number and the population size are both set to be 100 and N, respectively. The tournament selection operator, simulated binary crossover (SBX) and polynomial mutation operators [2] are used. The crossover probability used is 0.9 and the mutation probability is 1/6. We use the distribution indices [3] for crossover and mutation operators as $\eta_c = 20$ and $\eta_m = 20$, respectively. For statistical accuracy, 100 runs were performed and the average of the objective function value was taken. For simplicity, we take the same values of A, B and λ for all the pedestrians.

4.3 Results

The next three table gives the different set of optimal values of the parameters A, B and λ. As can be seen the solutions are multi-modal in nature. It is important to highlight that all the three solutions were found in the same simulation run.

The following summarizes the important results from and application of the evolutionary algorithm to this problem.

- Multi-modality is observed, i.e., there seem to be different parameter set giving almost the same fitness value. The use of evolutionary algorithms in such conditions is thus more justified. This is because these population based algorithms can find *all* or *many* optimal solutions in a single run.
- The A (strength term) and B (decay term) for German conditions are *more* and *less* than the corresponding Indian conditions, respectively. This has to do with cultural differences and the *personal space* is more in Germans than in Indians. Hence the A, B parameters are such that the repulsive force is more in German conditions than in Indian ones.
- The anisotropy parameter λ is *more* for Indians than Germans. This might be due to that Indians being less sensitive to pedestrians on sides. Among the differences in other parameters like A and B, the difference in λ, between Indians and Germans is more pronounced.
- The differences between the Indians and Germans and their walking behavior is reflected in the $h - v$ equations (see Equations 14 and 15). The evolutionary algorithm is able to *translate* this in the A and B parameters. This is important since with the optimal set of A and B values one could simulate a realistic pedestrian walking behavior, for both these countries.

5 Conclusions

While there have been many attempts to have a better microscopic pedestrian interaction model, none of them seem to model all the aspects considered in this

paper. This paper is an attempt to have a better model *without* any increase of number of parameters. Although not reported in this short paper, the new model also shows all the self-organizing phenomena that is observed in pedestrian traffic. We have presented an evolutionary algorithm based approach for calibration of this highly nonlinear and coupled interaction model. It came out that the problem of calibration is multi-modal in nature and many solutions were found in a single simulation run. We believe that evolutionary algorithms have a great potential in optimization of (nonlinearly coupled) pedestrian interactions and hope this study will simulate further research on parameter calibration studies for other countries/ behaviors.

References

1. Chattaraj, U., Seyfried, A., Chakroborty, P.: Comparison of pedestrian fundamental diagram across cultures. Adv. Complex Systems 12, 393–405 (2009)
2. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley & Sons Ltd., Chichester (2001)
3. Deb, K., Agarwal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
4. Helbing, D.: A mathematical model for the behavior of pedestrians. Behavioral Science 36, 289–310 (1991)
5. Helbing, D.: Verkehrsdynamik. Springer, Berlin (1997)
6. Helbing, D.: Traffic and related self-driven many-particle systems. Review of Modern Physics 73, 1067–1141 (2001)
7. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature 407, 487–490 (2000)
8. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Review E 51, 4282–4286 (1995)
9. Hughes, R.: A continuum theory for the flow of pedestrians. Transportation Research B 36, 507–535 (2002)
10. Johansson, A., Helbing, D., Shukla, P.: Specification of the social force pedestrian model by evolutionary adjust- ment to video tracking data. Adv. Complex Systems 10, 271–288 (2007)
11. Lewin, K.: Field Theory in Social Science. Harper & Brothers, New York (1951)
12. Okazaki, S.: A study of pedestrian movement in architectural space, Part 1: Pedestrian movement by the application on of magnetic models. Trans. of A.I.J. (283), 111–119 (1979)
13. Shukla, P.K.: Modeling and Simulation of Pedestrians. Masters thesis, Indian Institute of Technology Kanpur, India (2005)
14. Tilch, B., Helbing, D.: Evaluation of single vehicle data in dependence of the vehicle-type, lane, and site. In: Helbing, D., Herrmann, H., Schreckenberg, M., Wolf, D. (eds.) Traffic and Granular Flow 1999, pp. 333–338. Springer, Berlin (2000)
15. Yu, W.J., Chen, R., Dong, L.Y., Dai, S.Q.: Centrifugal force model for pedestrian dynamics. Phys. Rev. E 72(2), 026112 (2005)

Revising the Trade-off between the Number of Agents and Agent Intelligence

Marcus Komann[1] and Dietmar Fey[2]

[1] Friedrich-Schiller-University Jena
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
marcus.komann@googlemail.com
[2] University of Erlangen-Nürnberg
Martensstrasse 3, D-91058 Erlangen, Germany
dietmar.fey@informatik.uni-erlangen.de

Abstract. Emergent agents are a promising approach to handle complex systems. Agent intelligence is thereby either defined by the number of states and the state transition function or the length of their steering programs. Evolution has shown to be successful in creating desired behaviors for such agents. Genetic algorithms have been used to find agents with fixed numbers of states and genetic programming is able to balance between the steering program length and the costs for longer programs. This paper extends previous work by further discussing the relationship between either using more agents with less intelligence or using fewer agents with higher intelligence. Therefore, the Creatures' Exploration Problem with a complex input set is solved by evolving emergent agents. It shows that neither a sole increase in intelligence nor amount is the best solution. Instead, a cautious balance creates best results.

1 Introduction

"Emergence" can be shortly described as the *global* effect that is created by (inter)actions of *local* individuals [18]. Two classic examples for it are ant colonies, which are able to find the shortest path to a food resource with high probability [7] [2], or the "Game of Life", a special Cellular Automaton [19]. Emergence has been in the scope of researchers for some time and is still one of the major issues today [16]. This especially holds for rather complex systems because it might offer large benefits as result of the emergent process, e. g., self-organization, self-healing, self-optimization, self-configuration, and so on [3].

Usual "top-down" or "bottom-up" solution finding strategies do not work anymore for complex systems. The need for a different approach was discussed in [6], [1], or [18]. These authors agree that this new approach should rely on stepwise refinement of solutions in order to find a good or optimal solution in the end. Evolution of emergence has already been proposed by these authors as a way to achieve desired system capabilities.

Komann [11] evolved finite state machines that steer agents for the so-called "Creatures' Exploration Problem" (CEP), in which one or more agents shall visit

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 31–40, 2010.

a maximal number of non-blocked cells in a regular grid. The CEP is a special kind of a Cellular Automaton and forms the same highly nonlinear landscapes in the search space. Practical applications of this problem are cleaning robots that move through rooms of a university, the "Lawnmower Problem" [14], or the controlled movement of so-called "Marching Pixels" agents, which emergently traverse images in order to retrieve information about image objects [12].

For a specific set of input grids, Komann first showed that evolution can robustly create CEP agents that visit 80%-90% of possible cells. This is worse than optimal agents, which visit 96%-99%. But evolution found its results in 36 hours while searching optimal agents by enumeration and simulation needed 45 days (see the works of Halbach [10] [8]). Komann concluded that using evolution thus is feasble if good agents are needed fast.

In another work, Komann then tried to figure out the relationship between the amount of agents and the amount of states [13]. He showed that each approach on its own can be successful in principle but that a combination of both yields best results. However, the input set Komann used showed to be too small and simple. All evolutions with larger numbers of agents as well as larger numbers of states found agents that robustly visited 100% of free cells. The statements and conclusions from [13] are hence not very significant. In this paper, the same experiments are thus executed with a larger input set and the results are revised.

The results are especially interesting if it is possible to execute agents in parallel. Multiple agents have the capability to find better results faster. Anyhow, multiple agents cost multiple implementation effort. But less agents with higher capabilities are also costly. For the CEP model, the cost for an agent with a larger numbers of states is high concerning its execution in functional units. It might thus be useful to have more but cheaper parallel agents, which fulfill the task as well or even better than fewer agents with more states.

The paper is structured as follows. Sec. 2 describes the CEP and its state machines in more detail. In Sec. 3, implications of increased agent numbers and intelligence are discussed. Sec. 4 then recapitulates details on the evolutionary process. This is followed by the results of the evolution runs on the extended input in Sec. 5 before the last section summarizes the paper and gives an outlook to future work.

2 The Creatures' Exploration Problem

The "Creatures' Exploration Problem" (CEP) was defined in [9]: Given is a two-dimensional cellular field (regular grid) consisting of blocked cells and non-blocked cells. This grid serves as a fixed environment for agents that move around stepwise and have to fulfill a certain task. This task is visiting all non-blocked cells at least once in shortest time. An agent can look ahead one cell in the direction it currently moves in. It is only allowed to move onto non-blocked cells and can perform four different actions:

- `L` (turn left),
- `R` (turn right),
- `Lm` (move forward and then turn left),
- `Rm` (move forward and then turn right).

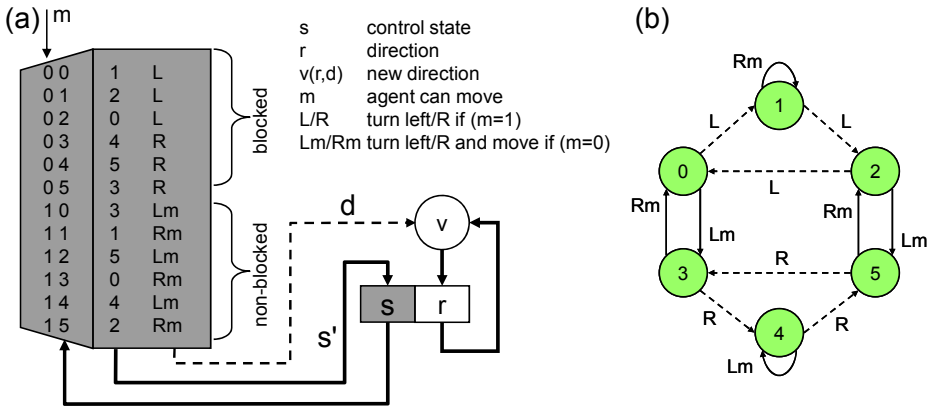


Fig. 1. A state machine (a) models an agent's behavior. Corresponding 6-state graph (b), dashed line for $m = 0$, solid line for $m = 1$.

The agent always tries to move forward. If the *front cell* (the cell ahead in the moving direction) is blocked, action `L` or `R` is performed. Action `Lm` or `Rm` is performed if the front cell is not blocked.

Behavior of an agent is modeled by a finite state machine (see Fig. 1). The state machine has n states $s \in \{0, 1 \ldots, n-1\}$, one binary input m (blocked or non-blocked front cell) and one binary output d (turn left or right). It can be depicted by a state graph (Fig. 1 (b)). E. g., being in current state $s = 3$ and receiving input $m = 0$ (do not move), this agent will perform the action R and the state transition to the next state $s' = 4$.

The amount of state machines that can be coded is $M = (n * \#y)^{(n*\#x)}$ where n is the number of states, $\#x$ is the number of different input values, and $\#y$ is the number of different output actions. In this paper, we look at state machines with maximally eight states. Thus, $16^{16} = 18,446,744,073,709,551,616$ possible behaviors can be defined. Note that not all agent behaviors described by state machines are distinct (e. g., permutation of the states leads to equivalent behaviors) or useful (e. g., state graphs that make little use of the inputs or are weakly connected).

The global behavior of a Cellular Automaton is difficult to foresee if one changes the local cell rules. The same holds for the CEP. If agents are executed with CEP state machines, changing one of the operations or switching to a different state often creates completely different agent movement on the same input grids. Combined with the exponential increase in number of state machines, this results in extremely large non-linear landscapes in the CEP's search space.

3 Increasing Agent Capabilities

The goal of this paper is to revise how well the CEP can be solved if we make agents more powerful. We use two different ways to give the agents more capabilities and

combine them. The first one is increasing the number of states per agent. The second one is increasing the number of agents.

Concerning the number of states, the simplest case is a one-state CEP agent. Such an agent can only decide if its front cell is blocked, move or not, and then turn to either side. In the worst case, this agent just moves and turns left all the time, eventually running in a four-cell-cycle instead of visiting more parts of a grid. A higher quantity of states allows the agent to have a more distinct movement scheme. If an agent is given more states, it has the opportunity to change states and thus directions in order to leave cycles and have a much higher rate of visited cells. This statement is not only true when turning from one state to two. It also holds for higher numbers of states because ever more states give the agent ever more opportunities to "react" on more sophisticated grids and also to leave larger cycles.

The second way of elevating agent power is by increasing the number of agents. This is a big extension because two agents might possibly visit the double amount of free cells than a single agent in the same time. A comparison of them is hence a little unfair. In order to decrease this unfairness, we let multiple agents start at the same position like single agents. No two agents can be in the same cell at the same time. Thus, the first agent is set to the same position as a single agent. A second agent's initial position is directly adjacent to the first one as is the third's and so on. Every added agent is as close to the first as possible in our tests. Other strategies would be to distribute multiple agents over the grid following a distribution function, e. g. uniform distribution or randomly, or to start the agents maximally far apart from each other. But those schemes would increase the unfairness because it would be much easier for multiple agents to visit all cells if they started far apart.

The application of multiple agents also raises the question for communication. In the CEP model, agents look ahead if the cell they want to move to is blocked or not. If one agent is used, the agent only sees a blocked cell in front if that front cell is blocked by the grid and can not be visited at any time. In order to decrease the mentioned unfairness when applying more agents, we want multiple agents to communicate as few as possible. Thus, the agents don't directly transfer knowledge between each other. The only communication occurs if agent A sees agent B in its front cell. A is then not allowed to move to the position of B and perceives that cell as blocked as long as B stays there.Using more agents should make sophisticated agent movements possible, too. Multiple agents should be able to leave cycles and thus have a higher cell visit rate than fewer agents. This is because an agent in a cycle can possibly find a cell of its cycle suddenly blocked by another agent, what makes it change its direction and leave the cycle.

4 Experimental Setup and Evolutionary Parameters

Komann evolved agents that solve the Creatures' Exploration Problem (CEP) [13] for the 26 input grids proposed by Halbach [8]. We want to do the same but with a larger set of 62 input grids. In order to make the results comparable, we

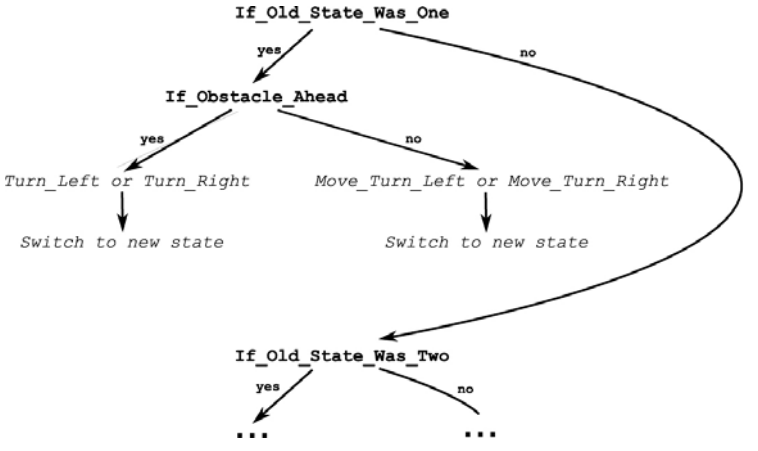


Fig. 2. Structure of a program/the state machine of an agent

have to use the same experimental setup like Komann. In this section, we hence shortly recapitulate details of the evolutionary processes.

A CEP agent behavior is represented by a state machine/automaton, which is executed by one or more agents on a grid. Such agents can also be steered by a small program consisting of several basic operations in a specific structure as presented in Fig. 2. The length of the program is depending on the number of states an agent is provided with. The evolution is allowed to change the operations that are depicted in gray italics. Operations in black stay fixed. Evolution is only allowed to change operations to an operation from that operation's family, for example `New_State_Is_One` can be changed to `New_State_Is_Three` but not to `Turn_Left` or `If_Old_State_Was_Four`.

One evolution run consisted of maximal 1000 generations with a population size of 10. The initial population was filled with random agent programs. Selection was done by the "Roulette Wheel" operator where the fitness of an individual defines the size of its segment on a virtual roulette wheel relative to the sum of the fitness of all individuals [15]. Selected individuals were replicated to a new generation with a probability of 30% in order to create some kind of elitist process. The reason for this strategy was the highly nonlinear nature of the search space. Without using an elitist strategy, the probability of leaving landscapes that promise high cell visit rates was too large. This is supported by Deb [4] [5], who showed that using elitist strategies can avoid leaving good local landshapes. Two selected individuals were recombined with a probability of 40% using one-point recombination.The remaining 30% of the population were mutated. The amount of mutated operations as well as their position in the chromosome were randomly chosen under the previously described structure constraints.

The fitness function represents the two objectives when evolving state machines for the CEP. The first one is that a maximal number of cells shall be

visited. The second, minor one is speed of the agents. Hence, the fitness F of an individual I was defined as:

$$F(I) = \sum_{k=grid_{first}}^{grid_{last}} \left(1000 * Cells_{visited}(k) - Speed(k)\right).$$

For $Speed(k)$, agents count the amount of steps they run on a grid according to their state machine. When all non-blocked cells have been visited, the number of steps is added to the global speed term. The maximum number of steps agents are allowed to take is 5000. When 5000 is reached, not all non-blocked cells are visited. Then, the lower visit number is added to the fitness via the $Cells_{visited}(k)$ term and 5000 is added via the speed term.

5 Results

The evolution results for 26 original input grids used by Komann were promising [13]. But the drawback of these tests was that too many automata visited all non-blocked cells for higher amounts of agents and states and significant statements could not be made. Hence, we repeated the tests with the same 26 input grids plus 36 new ones using the same operators, population sizes, generations, and probabilities. More input makes it more difficult for agents to be equally well in all grids because the decisions they make in a certain local situation in one grid might be bad on global scale in the same situation in another grid. The resulting success and speed tables should hence be more significant.

Using more input grids does not change the amount of possible automata. For a defined number of states, this amount is equal regardless of the number of input grids. The difference is that the automata are applied to more grids. Increasing the amount of input ever more will decrease the effectiveness of automata with a fixed number of states. The difference to less input lies mainly in the required time. The fitness of an individual is calculated by simulating all input grids and counting the number of visited pixels and required steps. Simulating more or larger input grids takes more time and so does the evolution.

The added grids were created to be closer to real-world problems than the initial grids. They consist of some letters and a set of different floor plans and starting positions. Especially the last 12 input grids were chosen to be difficult for the agents. The agents have to follow sophisticated paths to arrive at the farthest non-blocked cells there.

5.1 Visit Rates

Tab. 1 presents the success of evolved agents on the extended set of 62 inputs. It can be seen that, indeed, fewer automata reached the full visit rate of $13,576$ cells. Robust, full visits with $\sigma = 0$ required eight agents with five states (minimizing states) and five agents with seven states (minimizing agents). Besides, many of the automata with higher amounts of states and/or agents were very effective again. Most only slightly missed 100% visits with $\sigma = 0$.

Table 1. Average $\bar{x}$ and standard deviation σ of visited cells of the best individual of each of the five runs for 62 input images

	States																	
	1		2		3		4		5		6		7		8		Ø	
Agents	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
1	**2172**	0.0	**9660**	0.0	**9973**	548.7	**9807**	135.9	**10378**	571.0	**10463**	431.3	**10085**	498.9	**10502**	331.2	**9130**	314.6
2	**5704**	0.0	**11211**	2.204.4	**12552**	155.4	**12727**	199.9	**12701**	106.0	**12852**	157.7	**12826**	213.5	**12845**	206.5	**11677**	405.4
3	**7183**	0.0	**13334**	37.3	**13370**	38.4	**13373**	36.4	**13426**	23.8	**13388**	71.4	**13430**	37.4	**13421**	27.4	**12616**	34.0
4	**8300**	0.0	**12776**	1.495.4	**13528**	39.9	**13546**	18.4	**13564**	11.3	**13571**	0.6	**13568**	5.4	**13570**	1.6	**12803**	196.6
5	**8785**	0.0	**12952**	1.177.8	**13564**	16.1	**13566**	4.4	**13567**	8.4	**13573**	6.0	**13576**	0.0	**13572**	6.0	**12894**	152.3
6	**9500**	0.0	**13572**	2.9	**13574**	1.4	**13574**	3.5	**13575**	0.4	**13575**	1.2	**13575**	1.2	**13576**	0.0	**13065**	1.3
7	**9921**	0.0	**13574**	0.8	**13574**	0.5	**13575**	0.8	**13575**	1.0	**13576**	0.0	**13576**	0.0	**13576**	0.0	**13118**	0.4
8	**10270**	0.0	**13575**	0.7	**13575**	1.0	**13576**	0.8	**13576**	0.0	**13576**	0.0	**13576**	0.0	**13576**	0.0	**13162**	0.3
Ø	**7729**	0.0	**12582**	614.9	**12964**	100.2	**12968**	50.0	**13045**	90.2	**13072**	83.5	**13027**	94.5	**13080**	71.6	**12308**	138.1

Table 2. Average $\bar{x}$ and standard deviation σ of speed of the best individual of each of the five runs for 62 input images

	States																	
	1		2		3		4		5		6		7		8		Ø	
Agents	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
1	**310000**	0	**217744**	2097	**219742**	3161	**232920**	10866	**222133**	3492	**216729**	8984	**225873**	16701	**213441**	8506	**232323**	6726
2	**310000**	0	**163374**	66037	**130067**	6661	**139777**	14102	**124408**	6376	**141342**	17918	**127429**	10474	**131631**	11416	**158503**	16623
3	**285527**	0	**98324**	7929	**95132**	8100	**95765**	7529	**93796**	9777	**95567**	4442	**93619**	3015	**91341**	5889	**118634**	5835
4	**262993**	0	**112877**	85277	**66107**	8123	**66561**	4030	**61797**	2789	**62373**	563	**62864**	2595	**61548**	2530	**94640**	13238
5	**257930**	0	**90239**	67756	**53855**	5664	**48042**	4092	**50941**	5577	**43177**	1022	**42996**	1334	**44549**	1741	**78966**	10898
6	**251650**	0	**42934**	6336	**41375**	4160	**39265**	7040	**42230**	4798	**33468**	3753	**32064**	2925	**29911**	1573	**64112**	3823
7	**251107**	0	**33080**	3592	**33237**	3783	**33491**	2028	**32261**	3381	**25553**	2070	**24967**	1626	**25256**	1121	**57369**	2200
8	**224018**	0	**31816**	1445	**28304**	3511	**26211**	4527	**22041**	1492	**21075**	340	**20274**	1012	**20262**	1080	**49250**	1676
Ø	**269153**	0	**98798**	30059	**83477**	5395	**85254**	6777	**81201**	4710	**79911**	4887	**78761**	4960	**77242**	4232	**106725**	7627

However, the conclusions made by Komann for 26 inputs grids are mirrored in Tab. 1. Although the 62 input grids were a lot more difficult, the evolution was successful and robust in higher amount of states and agents. The standard deviation was again higher in lower amounts of states and agents. This indicates that even this large and sophisticated set of input grids does not exceed the capabilities of eight agents with eight states. Even some more input grids could be learned and successfully traversed by agents with these amounts. Full visit rates were already possible with a small amount of states and agents. Sole increase of either states or agents was not effective. Like stated in [13], both values had to be increased simultaneously to achieve good results.

Besides the similarities, two differences between the results for 26 and 62 input grids showed up. The first one is the influence of either states or agents. For 26 inputs, increasing agents was more effective than increasing states. Now, the average values in the bottom row and the right column are similar. They indicate no significant difference in visiting success of either of the variables. The second difference is that eight states here were slightly more successful than seven states. Using more input, the surprising effect of seven states being better than the (in principle) more powerful eight states vanished. For more complex input, the larger search space of more states seems to be more effective.

5.2 Speed of the Agents

Tab. 2 illustrates the number of steps required by the agents to visit their amount of cells in Tab. 1. 5,000 steps were maximally allowed per grid. Thus, $62 * 5,000 = 310,000$ steps could maximally be taken.

Like in the success table, the tendencies from the tests with 26 input grids mirror here. Increasing both amount of states and amount of agents improved the results significantly. Only using more agents (right column) again improved the speed in every agent addition. A little different to Komann's results for 26 grids where (apart from $states = 1$) increasing the states did not speed up the agents, a continuous decrease in required steps can be seen here while increasing states (bottom row).

The σ values increased slightly. Compared to the increase in possible steps (130,000 to 310,000), their average increase was smaller. They show that the evolution results were robust because they were relatively small with an overall average of $\sigma = 7,627$ for a steps average of $\bar{x} = 106,725$, which is just 7.1%.

6 Summary and Outlook

Complex systems consisting of locally acting parts that evoke global system characteristics mostly can't be designed in classic ways. This is due to the often unforeseeable, nonlinear behavior emerging on global level that results from changing the local units. The same reason creates the highly nonlinear search spaces that have to be traversed to optimize these global system characteristics.

In this paper, we discussed how well evolved emergent agents can solve the global objective to visit all non-blocked cells fast in several grids. We especially

revised the question if it is better to use more agents or to give the agents more intelligence. The tests showed that the latter already yields good results but that more agents perform even better. Anyhow, the tests hinted that mixing both seems to be the best way for agents of the Creatures' Exploration Problem.

Which option should be used in a specific application depends on the implementation costs for either option. Using more agents as well as using more states normally costs hardware resources. In the case of massively-parallel fine-grained models like Cellular Automata [19], Cellular Neural Networks [17], or Marching Pixels [12], implementing more agents that work simultaneously is no problem. It is rather a feature of these models that they comprise multi-agent support. When realizing such models in hardware, it is expensive to make the single cells more powerful because they require more functional units, e. g., more memory to save the states and a larger state transition table. But multiple agents in some sense come for free in these models. The problem there is the search for correct local rules that result in desired global behavior. As Komann stated, for such systems, evolving agent behavior seems to be a feasible approach.

Future work on this topic consists of extending the communication capabilities of the agents and then comparing the results to those in this paper. Extended communication should make multiple agents even more powerful. This could, for example, mean giving the agents small memory where they could memorize their last few steps. This information could then be communicated from one agent to another in order to prevent the receiving agent from visiting the same cells again. Another opportunity that would especially fit to the mentioned fine-grained parallelism would be to save information in the cells if any agent already visited a cell before. This would refer very much to animals searching and eating food on a plain. Agents then should visit fewer previously visited cells.

References

1. Anderson, C.: Creation of desirable complexity: strategies for designing self-organized systems. In: Complex Engineered Systems, pp. 101–121. Perseus Books Group (2006)
2. Beckers, R., Deneubourg, J.L., Goss, S.: Trails and U-turns in the selection of the shortest path by the ant Lasius Niger. Journal of Theoretical Biology 159, 397–415 (1992)
3. Branke, J., Schmeck, H.: Evolutionary design of emergent behavior. In: Organic Computing, March 2008, pp. 123–140. Springer, Heidelberg (2008), http://www.springer.com/978-3-540-77656-7
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
5. Deb, K.: A robust evolutionary framework for multi-objective optimization. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 633–640. ACM, New York (2008)
6. Edmonds, B.: Using the experimental method to produce reliable self-organised systems. In: Brueckner, S.A., Di Marzo Serugendo, G., Karageorgos, A., Nagpal, R. (eds.) ESOA 2005. LNCS (LNAI), vol. 3464, pp. 84–99. Springer, Heidelberg (2005)

7. Goss, S., Aron, S., Deneubourg, J., Pasteels, J.: Self-organized shortcuts in the argentine ant. Naturwissenschaften 76(12), 579–581 (1989), http://dx.doi.org/10.1007/BF00462870
8. Halbach, M.: Algorithmen und Hardwarearchitekturen zur optimierten Aufzählung von Automaten und deren Einsatz bei der Simulation künstlicher Kreaturen. Ph.D. thesis, Technische Universität Darmstadt (2008)
9. Halbach, M., Heenes, W., Hoffmann, R., Tisje, J.: Optimizing the behavior of a moving creature in software and in hardware. In: Sloot, P.M.A., Chopard, B., Hoekstra, A.G. (eds.) ACRI 2004. LNCS, vol. 3305, pp. 841–850. Springer, Heidelberg (2004)
10. Halbach, M., Hoffmann, R., Both, L.: Optimal 6-state algorithms for the behavior of several moving creatures. In: El Yacoubi, S., Chopard, B., Bandini, S. (eds.) ACRI 2006. LNCS, vol. 4173, pp. 571–581. Springer, Heidelberg (2006)
11. Komann, M., Ediger, P., Fey, D., Hoffmann, R.: On the effectiveness of evolution compared to time-consuming full search of optimal 6-state automata. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 280–291. Springer, Heidelberg (2009)
12. Komann, M., Fey, D.: Realising emergent image preprocessing tasks in cellular-automaton-alike massively parallel hardware. International Journal of Parallel, Emergent and Distributed Systems 22(2), 79–89 (2007)
13. Komann, M., Fey, D.: Evaluating the evolvability of emergent agents with different numbers of states. In: GECCO, pp. 1777–1778. ACM, New York (2009)
14. Koza, J.R.: Scalable learning in genetic programming using automatic function definition, pp. 99–117 (1994)
15. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, London (1996)
16. Müller-Schloer, C., Sick, B.: Emergence in Organic Computing systems: Discussion of a controversial concept. In: Yang, L.T., Jin, H., Ma, J., Ungerer, T. (eds.) ATC 2006. LNCS, vol. 4158, pp. 1–16. Springer, Heidelberg (2006)
17. Roska, T., Chua, L.: The cnn universal machine: an analogic array computer. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 40(3), 163–173 (1993)
18. Wolf, T.D., Holvoet, T.: Emergence versus self-organisation: Different concepts but promising when combined. In: Brueckner, S.A., Di Marzo Serugendo, G., Karageorgos, A., Nagpal, R. (eds.) ESOA 2005. LNCS (LNAI), vol. 3464, pp. 1–15. Springer, Heidelberg (2005)
19. Wolfram, S.: A New Kind of Science. Wolfram Media Inc., Champaign (2002)

Sexual Recombination in Self-Organizing Interaction Networks

Joshua L. Payne and Jason H. Moore

Computational Genetics Laboratory, Dartmouth Medical School,
Lebanon, NH 03756, USA
Joshua.L.Payne@Dartmouth.edu

Abstract. We build on recent advances in the design of self-organizing interaction networks by introducing a sexual variant of an existing asexual, mutation-limited algorithm. Both the asexual and sexual variants are tested on benchmark optimization problems with varying levels of problem difficulty, deception, and epistasis. Specifically, we investigate algorithm performance on Massively Multimodal Deceptive Problems and NK Landscapes. In the former case, we find that sexual recombination improves solution quality for all problem instances considered; in the latter case, sexual recombination is not found to offer any significant improvement. We conclude that sexual recombination in self-organizing interaction networks may improve solution quality in problem domains with deception, and discuss directions for future research.

1 Introduction

Many natural and physical systems can be characterized as networks, where vertices denote system components and edges denote component interactions. Recent advances in computational power and the increased availability of large-scale data sets have provided several novel insights regarding the influence of network structure on the functionality and vulnerability of these systems [15], and on the dynamical processes that occur within them [4]. For example, at the cellular level, the average connectivity of a genetic regulatory network affects both its ability to discover novel phenotypes and its robustness to perturbation [3]; at the population level, the spatial localization of interaction events affects the maintenance of genetic diversity [12,22], the evolution of cooperation [16,10], and the emergence of altruistic behavior [25].

Inspired by the interaction networks of such complex adaptive systems, recent advances in cellular evolutionary algorithms have employed heterogeneous interaction networks as population structures. The saturation dynamics of advantageous alleles have been investigated in both small-world [7] and scale-free [7,19] interaction networks, and scale-free population structures have been analyzed in the context of genetic algorithms for single [6] and multiobjective optimization problems [13,14], and in evolutionary robotics applications [5].

While such complex population structures bear a closer resemblance to some natural systems than their lattice-based predecessors [24,23,8], the analogy only

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 41–50, 2010.

goes so far. In all of the examples cited above, the interaction network was generated prior to the evolution of the population, and its structure was held fixed throughout the evolutionary process. Though this approach provides useful insights regarding the influence of various topological properties on evolutionary search [19], it is a gross oversimplification of the interaction networks of natural systems, which are both dynamic and self-organizing. To date, only two studies have attempted to incorporate these salient features of biological systems into cellular evolutionary algorithms [2,26]. In [2], the global ratio of the horizontal and vertical dimensions of a two-dimensional lattice population structure was adaptively altered via a feedback loop with the evolutionary dynamics of the population, offering significant performance improvements over static lattice-based interaction networks. However, the local neighborhood structure of the interaction network was assumed *a priori* and held constant throughout the run and the global population structure was assumed to be regular. In contrast, the algorithm presented in [26] is not only dynamic, but also allows for the self-organization of irregular local neighborhood structures and the emergence of global network properties similar to the complex adaptive systems from which nature-based optimization algorithms draw their inspiration. The results of [26] demonstrate that mutation-limited genetic algorithms structured on such self-organizing interaction networks improve both diversity maintenance and solution quality over panmictic and lattice-based population structures, in specific problem domains.

Here, we extend the algorithm presented in [26] to include sexual recombination, a variation operator that is integral to the exploration phase of evolutionary search. We compare the asexual and sexual variants of this algorithm on benchmark optimization problems with varying levels of problem difficulty, multimodality, and epistasis.

2 Methods

2.1 Self-Organizing Interaction Networks

The details of the asexual variant of the self-organizing interaction networks considered in this study are provided in [26]. For completeness, we provide a high-level overview of this algorithm here.

The interaction network is initialized as a ring of M vertices, where each vertex connects to its two nearest neighbors. A population of size M is then randomly initialized, and each individual is placed in its own vertex. Since vertices are always occupied by a single individual, the terms vertex and individual will be used interchangeably.

The coupled evolution of the population and the interaction network is broken into two phases per generation. In the first phase, M parents are selected to produce offspring, each of which is placed in a new vertex. This results in a network with $2M$ vertices. An offspring vertex connects to its parent vertex, and inherits each of its parent's edges with probability p_{add}. If an offspring inherits an edge from its parent, then the parent loses that edge with probability p_{remove}.

Since offspring connect to their parents, and a parent can only lose an edge if its offspring gains that edge, it is guaranteed that the network will remain connected (i.e., a finite path exists between all vertices). Parents are selected with uniform probability from the population, with replacement. Thus, parent selection does not depend on fitness. Offspring faithfully inherit the parental genotype, subject to mutation.

In the second phase, selection pares the interaction network back down to M vertices, as follows. A vertex is selected at random with uniform probability from the population. This vertex then competes with its lowest-fitness neighbor. Of this pair, the lower fitness vertex is removed from the network and the higher fitness vertex inherits all of its links. Since the individual with the highest fitness cannot lose one of these competitions, elitism is implicitly included in this framework.

A key insight of [26] is that the fitness of a vertex should be considered in the context of its local neighborhood. To do this, each individual is given a rank r based on how its raw fitness f compares with the raw fitnesses of its d neighbors. A rank of $r = 0$ denotes the best individual in a neighborhood and a rank of $r = d$ denotes the worst. Based on the individual's rank r, it is assigned a contextual fitness f' according to

$$f' = \frac{d - r}{d}, \tag{1}$$

which is used in all competition events. After an individual is removed from the network, the contextual fitnesses of all its neighbors are reevaluated.

These two phases are repeated for a specified number of generations. Each vertex addition and removal transforms the network away from its original homogeneous form to something more closely resembling the topologies of complex, natural systems [26] (Fig. 1).

2.2 Sexual Recombination

Here, we propose a simple extension of the asexual reproduction phase considered in [26]. When a vertex i is chosen for reproduction, a mate j is subsequently selected at random from the neighborhood of vertex i. A recombination operator is then applied to the individuals situated in vertices i and j to form an offspring in vertex z, which attaches to i and inherits i's links in the same manner as the asexual variant.

2.3 Benchmark Problems

In this section, we briefly describe the two benchmark optimization problems used in this study. These problems were chosen because they possess important characteristics of real optimization tasks, such as multimodality, deception, and epistasis, and allow for a tunable degree of problem difficulty. Due to space constraints, we limit our description of these problems to their defining characteristics and the details needed to replicate our experiments. The interested reader should refer to [2,11] for more details.

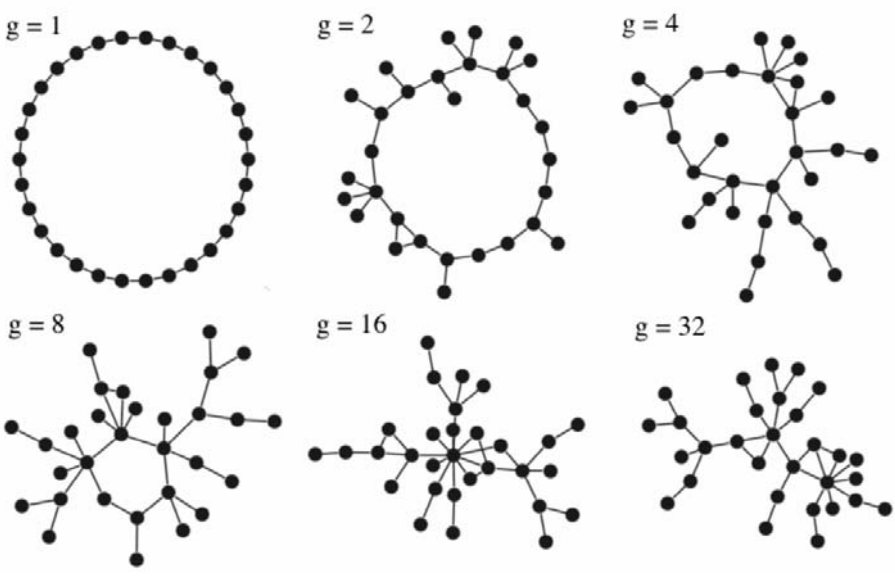


Fig. 1. Visualization of the evolutionary dynamics of self-organizing interaction networks. The network is initialized as a ring and the vertex addition and removal events of subsequent generations (g) transform the network into a heterogenous form. For visual clarity, these networks are deliberately smaller than any used in the experiments ($M = 30$ vertices). These networks were generated by evolving a population on NK landscapes with $N = 30$ and $K = 2$ (see Section 2.3).

Massively Multimodal Deceptive Problems. Massively multimodal deceptive problems [2,9] consist of k concatenated subproblems, each with two global optima and a single deceptive suboptimum. The subproblems are six bits long, and the fitness contribution of each subproblem depends on the unitation of the bits. Specifically, the unitation is used as an index into the vector $\langle 1.00, 0.00, 0.36, 0.64, 0.36, 0.00, 1.00\rangle$. Thus, the global optima of each subproblem are at maximum hamming distance from one another (at a unitation of zero and six) and provide a fitness contribution of one. The deceptive suboptimum provides a fitness contribution of 0.64 and is located at a unitation of three. Maximum fitness is k, which we normalize to a maximum of 1.

NK Landscapes. NK landscapes are abstract models of fitness surfaces [11]. Each of the N bits in a binary string epistatically interact with K neighboring bits to provide a fitness contribution. These contributions are in the range $(0, 1)$ and are extracted from a randomly generated look-up table with 2^{K+1} entries for each of the N bits. The ruggedness of the fitness surface increases with the number of interacting bits K. Fitness is defined as the arithmetic mean of the N fitness contributions.

2.4 Experimental Design

To be consistent with [26], we use a population size of $M = 100$, a mutation rate of $1/N$ (where N is the length of the binary string), and $p_{add} = p_{remove} = 0.1$. We use bit-flip mutation and, in the sexual case, one-point crossover (in addition to mutation). For the MMDP, we consider $20 \leq k \leq 640$ and allow evolution to proceed for 1000 generations. (Preliminary experimentation with maximum generation time indicated that 1000 generations was sufficient for population convergence to occur, for all values of k considered.) For each problem instance, we perform 500 independent replications, where the asexual and sexual variants are seeded with the same initial populations. For the NK landscapes, we consider $N = 30$ and $2 \leq K \leq 14$, and allow the population to evolve for 5000 generations, consistent with [26]. For each value of K, we randomly generate 500 problem instances. For each problem instance, we generate a random initial population and use it to seed both algorithm variants. This experimental design allows for a paired statistical comparison of all results.

3 Results

3.1 Massively Multimodal Deceptive Problems

For all instances of the MMDP considered, the asexual and sexual algorithm variants both converged on suboptimal, deceptive solutions. However, the sexual variant always found higher fitness solutions. For example, in Fig. 2a, we depict the average best fitness of the two algorithm variants on a representative MMDP instance, with $k = 80$. In this case, the average best solution found by the asexual variant had a fitness of 0.687 ($\pm$0.0004 s.e.), whereas the average best solution found by the sexual variant had a fitness of 0.717 ($\pm$0.0005 s.e.).

This trend was consistent across all MMDP instances. In Fig. 3, we depict the fitness of the best solutions found at the end of each replication as a function of problem difficulty, for the asexual (A) and sexual (S) algorithm variants. The average best fitness of the sexual variant was always significantly higher than the average best fitness of its asexual counterpart ($p < 0.05$, paired t-test). As problem difficulty increased, the average best fitness found by both algorithm variants decreased (Fig. 3), an expected result given that the population size was held constant for all experiments.

3.2 NK Landscapes

In Fig. 4, we compare the fitness of the best solutions found by the asexual (A) and sexual (S) algorithms on NK landscapes with $N = 30$ and varying levels of epistasis (K). For both algorithm variants, the fitness of the best solutions varied non-monotonically with K, such that the highest fitness was observed for $6 \leq K \leq 8$. For all values of K, the distributions of best fitness were statistically indistinguishable between the two algorithms ($p > 0.05$, paired t-test).

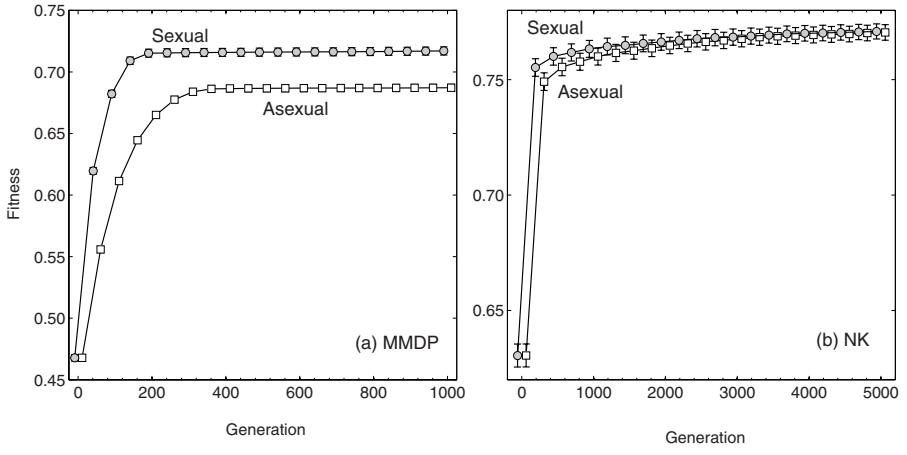


Fig. 2. Best raw fitness per generation of the sexual and asexual algorithm variants on the (a) MMDP ($k = 80$) and (b) NK ($N = 30, K = 10$) test problems. Data points are the average of all replications, and error bars denote standard error. (In (a) the error bars are smaller than the symbol size.) The data are deliberately offset in the horizontal dimension for visual clarity.

As an illustrative example, we depict in Fig. 2b the fitness of the best solutions found by the two algorithms as a function of generation number, for a representative case of intermediate epistasis ($K = 10$). On these problem instances, the best solutions found by both algorithm variants had an average fitness of 0.770 (± 0.004 s.e.).

4 Discussion

In this study, we have demonstrated that including sexual recombination in self-organizing interaction networks can improve solution quality in some problem domains. In a direct comparison between an asexual, mutation-limited algorithm [26] and a sexual, recombinative variant, we found that the sexual variant discovered higher quality solutions when the problem instance was deceptive, but offered no advantage when the problem instance was epistatic.

Our results on NK landscapes contrast those observed with panmictic genetic algorithms [1], where recombination was shown to offer an advantage over a mutation-limited variant for intermediate levels of epistasis ($12 < K < 32$ for NK landscapes with $N = 96$). We found that the asexual and sexual variants produced statistically indistinguishable results for all values of K considered. This result follows from the fact that both the ruggedness of a fitness landscape and the fitness difference between peaks and valleys increases monotonically with K [17]. Thus, not only is recombination more likely to produce offspring in maladaptive fitness valleys as K increases, but the sheer depth of these valleys acts to preclude any

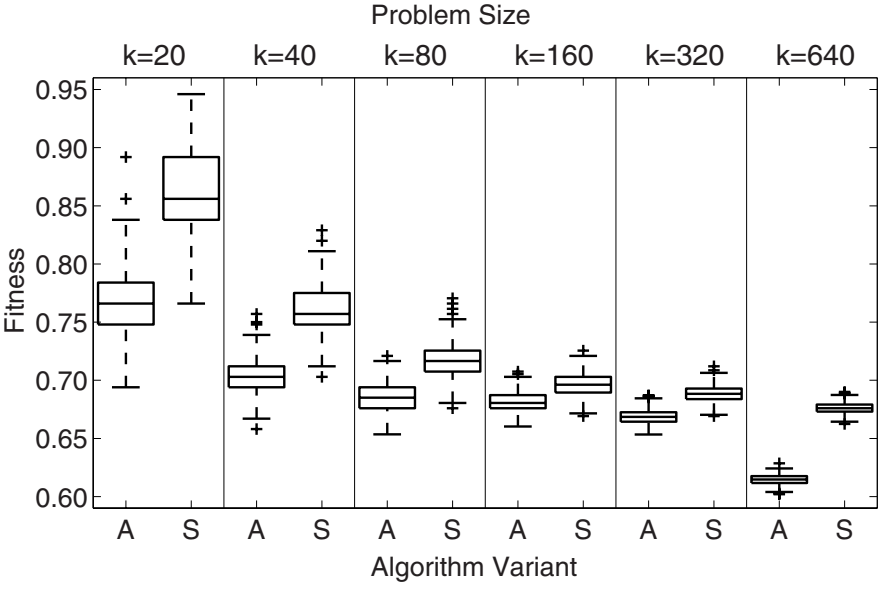


Fig. 3. Best raw fitness after 1000 generations for various MMDP problem sizes (k), using the asexual (A) and sexual (S) algorithm variants. The sexual variant produced significantly higher quality solutions in all cases ($p < 0.05$, paired t-test).

possibility of mutational escape. While such maladaptive movements in genome-space are often considered prerequisites for adaptive peak shifts in epistatic fitness landscapes [17], they were not found to offer any advantage here.

The coupling of population dynamics and population structure [26] makes these self-organizing interaction networks unique among those employed in cellular evolutionary algorithms, and allows for better diversity maintenance than static, lattice-based interaction networks [26]. An analysis of several structral properties of the interaction networks evolved in this study, including characteristic path length, clustering coeffecient, degree distribution, and assortativity, did not reveal any significant difference between the asexual and sexual cases. Further, we did not find any relationship between the structure of the evolved interaction networks and the underlying problem domain or its corresponding difficulty. Future work will seek to investigate the relationship between the topological characteristics of self-organizing interaction networks and the population-level distribution of genetic information, in order to better understand how diversity is maintained in both the asexual and sexual cases. This will complement previous analyses of population distributions in static, regular interaction networks [20].

Self-organizing population structures offer the potential to improve the efficacy of evolutionary search, and better mimic some of the features of complex biological systems. However, several important aspects of natural interaction

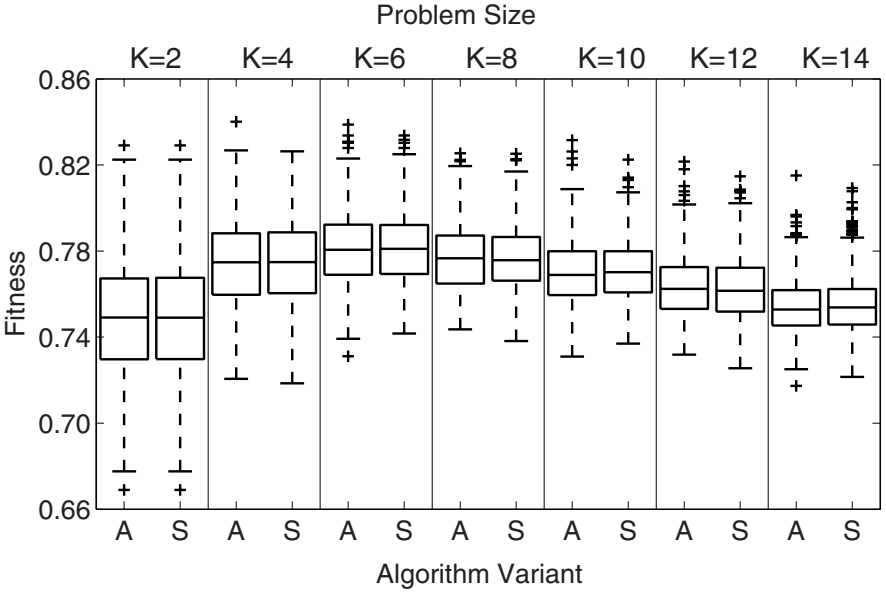


Fig. 4. Best raw fitness after 5000 generations as a function of the degree of epistasis K in NK landscapes with $N = 30$ for the asexual (A) and sexual (S) algorithm variants. Solution quality was statistically indistinguishable between the asexual and sexual algorithm variants ($p > 0.05$, paired t-test).

networks are not included in this framework. In biotic populations, individuals often choose their own social ties, as opposed to inheriting them from their parents, as was the case here. This 'active linking' has offered novel insights into such pertinent topics as the evolution of cooperation, in both theoretical [18] and experimental settings [21]. We believe that allowing individuals to strengthen (or weaken) their ties with competitors or mating partners may further improve the search capabilities of cellular evolutionary algorithms that employ self-organizing interaction networks. Current research is directed along these lines.

References

1. Aguirre, H.E., Tanaka, K.: Genetic algorithms on NK-landscapes: Effects of selection, drift, mutation, and recombination. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 131–142. Springer, Heidelberg (2003)
2. Alba, E., Dorronsoro, B.: The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Transactions on Evolutionary Computation 9(2), 126–142 (2005)

3. Aldana, M., Balleza, E., Kauffman, S.A., Resendis, O.: Robustness and evolvability in genetic regulatory networks. Journal of Theoretical Biology 245, 433–448 (2007)
4. Barrat, A., Barthélemy, M., Vespignani, A.: Dynamical Processes on Complex Networks. Cambridge University Press, Cambridge (2008)
5. Gasparri, A., Panzieri, S., Pascucci, F., Ulivi, G.: A spatially structured genetic algorithm over complex networks for mobile robot localisation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4277–4282. IEEE Press, Los Alamitos (2007)
6. Giacobini, M., Preuss, M., Tomassini, M.: Effects of scale-free and small-world topologies on binary coded self-adaptive cea. In: Gottlieb, J., Raidl, G.R. (eds.) Evolutionary Computation and Combinatorial Optimization, pp. 86–98. Springer, Heidelberg (2006)
7. Giacobini, M., Tomassini, M., Tettamanzi, A.: Takeover times curves in random and small-world structured populations. In: Beyer, H.G. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005, pp. 1333–1340. ACM Press, New York (2005)
8. Giacobini, M., Tomassini, M., Tettamanzi, A., Alba, E.: Selection intensity in cellular evolutionary algorithms for regular lattices. IEEE Transactions on Evolutionary Computation 9(5), 489–505 (2005)
9. Goldberg, D., Deb, K., Horn, J.: Massive multimodality, deception, and genetic algorithms. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving From Nature, pp. 37–46. North-Holland, Amsterdam (1992)
10. Hauert, C., Doebeli, M.: Spatial structure often inhibits the evolution of cooperation in the snowdrift game. Nature 428, 643–646 (2004)
11. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
12. Kerr, B., Riley, M.A., Feldman, M.W., Bohannan, B.J.M.: Local dispersal promotes biodiversity in a real life game of rock-paper-scissors. Nature 418, 171–174 (2002)
13. Kirley, M., Stewart, R.: An analysis of the effects of population structure on scalable multiobjective optimization problems. In: Thierens, D. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2007, pp. 845–852. ACM Press, New York (2007)
14. Kirley, M., Stewart, R.: Multiobjective optimization on complex networks. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 81–95. Springer, Heidelberg (2007)
15. Newman, M.E.J., Barabási, A.L., Watts, D.J. (eds.): The Structure and Dynamics of Networks. Princeton University Press, Princeton (2006)
16. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
17. Ostman, B., Hintze, A., Adami, C.: Impact of epistasis on evolutionary adaptation. arXiv:0909.3506v1 (2009)
18. Pacheco, J.M., Traulsen, A., Ohtsuki, H., Nowak, M.A.: Repeated games and direct reciprocity under active linking. Journal of Theoretical Biology 250, 723–731 (2008)
19. Payne, J.L., Eppstein, M.J.: Evolutionary dynamics on scale-free interaction networks. IEEE Transactions on Evolutionary Computation 13(4), 895–912 (2009)
20. Preuss, M., Lasarczyk, C.: On the importance of information speed in structured populations. 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. 91–100. Springer, Heidelberg (2004)
21. Rand, D.R., Dreber, A., Ellingsen, T., Fudenberg, D., Nowak, M.A.: Positive interactions promote public cooperation. Science 325, 1272–1275 (2009)

22. Reichenbach, T., Mobilia, M., Frey, E.: Mobility promotes and jeopardizes biodiversity in rock-paper-scissors games. Nature 448, 1046–1049 (2007)
23. Rudolph, G.: On takeover times in spatially structured populations: array and ring. In: Lai, K.K., Katai, O., Gen, M., Lin, B. (eds.) Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications, APGA-2000, pp. 144–151. Global Link Publishing Company, Hong Kong (2000)
24. Sarma, J., De Jong, K.: An analysis of the effect of neighborhood size and shape on local selection algorithms. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 236–244. Springer, Heidelberg (1996)
25. Werfel, J., Bar-Yam, Y.: The evolution of reproductive restraint through social communication. Proceedings of the National Academy of Science 101(30), 11019–11024 (2004)
26. Whitacre, J.M., Sarker, R.A., Pham, Q.T.: The self-organization of interaction networks for nature-inspired optimization. IEEE Transactions on Evolutionary Computation 12(2), 220–230 (2008)

Symbiogenesis as a Mechanism for Building Complex Adaptive Systems: A Review

Malcolm I. Heywood and Peter Lichodzijewski

Faculty of Computer Science, Dalhousie University
Halifax, NS, Canada
{mheywood,piotr}@cs.dal.ca

Abstract. In 1996 Daida *et al.* reviewed the case for using symbiosis as the basis for evolving complex adaptive systems [6]. Specific observations included the impact of different philosophical views taken by biologists as to what constituted a symbiotic relationship and whether symbiosis represented an operator or a state. The case was made for symbiosis as an operator. Thus, although specific cost benefit characterizations may vary, the underlying process of symbiosis is the same, supporting the operator based perspective. Symbiosis provides an additional mechanism for adaption/ complexification than available under Mendelian genetics with which Evolutionary Computation (EC) is most widely associated. In the following we review the case for symbiosis in EC. In particular, symbiosis appears to represent a much more effective mechanism for automatic hierarchical model building and therefore scaling EC methods to more difficult problem domains than through Mendelian genetics alone.

1 Introduction

Evolutionary Computation (EC) has long been associated with a Darwinian model of evolution in which natural selection represents a metaphor for performance evaluation and the motivation for maintaining a population of candidate solutions, whereas metaphors from Mendelian genetics are generally invoked to support the specifics of the representation and provide a model for credit assignment [11]. As such this mirrors the classical development of biology, with recent extensions including the introduction of developmental evolution – therefore reinforcing the use of Mendelian genetics – to the widespread use of coevolution, particularly cooperative and competitive models. Indeed, even calls for the use of more accurate biological models in EC have generally focused on the genetics, thus reinforcing discoveries such as the process of transcription [4]. Conversely, symbiosis as a coevolutionary process has been much less widely studied in the EC literature.

Symbiosis was defined by De Bary in 1879 as the living together of organisms from different species c.f., "unlike organisms live together" [8] (see [24,6,20] for current surveys of the concept). As such the process can encompass both exploitive parasitic relationships and co-operative mutualistic associations. However, central to symbiosis is a requirement for long-term, but not necessarily

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 51–60, 2010.

physical, association between partnering entities. The nature of the association, and therefore the degree of antagonism versus mutualism linking different partners, will vary as a function of environmental factors (see for example the closing commentary in [6]). When the symbiotic association leads to a long-term relationship that, say, converts an initially exploitive relationship into one of cooperative dependence *resulting in a new species* then the process is considered to be that of symbiogenesis [26,27,25].

The core components include: (1) partners entering in a relationship from different species/ organisms; (2) partners adapting phenotypically under selection pressure as a result of the symbiotic relationship, and; (3) a long term association which facilitates the creation of a new species of organism(s). The first two points are sufficient for symbiosis, whereas all three points provide symbiogenesis. The result is therefore increased functionality in the case of the final host entity, through the learning or application of traits developed independently by the symbiont(s) [26,27,25]. Conversely, a Darwinian model emphasizes the vertical inheritance of genetic variation through sexual reproduction of partners from the same species [19,25]. From an EC perspective symbiogenesis is a form of coevolution that has the potential to provide the basis for hierarchical/ component-wise model building; whereas competitive coevolution provides a mechanism for scaling EC to problem domains with truly vast state spaces and cooperative coevolution supports processes by which parallel problem decomposition / diversity is emphasized. Indeed systems utilizing multiple forms of coevolution are beginning to appear in EC (e.g., [38,21]), whilst the interaction of multiple evolutionary mechanisms in biology is widely acknowledged [19]. In the following we review biological properties of symbiosis – thus revisit a general abstract model of symbiosis that appears to be particularly useful under an EC context – as well as recent attempts to make use of symbiotic style algorithms in EC.

2 Biological Context

Despite De Bary's early recognition of symbiosis, it was not until the 1970's that the phenomena received more widespread recognition. In particular Lynn Margulis was instrumental in promoting Serial Endosymbiosis Theory as the mechanism by which evolution from prokaryote to eukaryote took place [23]–[27]. Moreover, most autopoietic entities require symbiotic (as well as Darwinian) models of development [25]. From the perspective of theoretical biology, John Maynard Smith was an early proponent, abstracting the concept of symbiogenesis as a mechanism by which complexity may be increased [28]. Figure 1 summarizes his model in which: Individuals (symbionts) from candidate species currently coexisting in a common genepool/ population/ ecosystem become enclosed within a 'compartment' such that a subset of individuals interact, thus identifying the partners. Over time, the interaction results in a mechanism being established for the replication of the partners i.e., the interaction is beneficial as measured by natural selection. Thus, the specific form of a symbiotic coevolutionary interaction was not emphasized, but rather the key factors were that

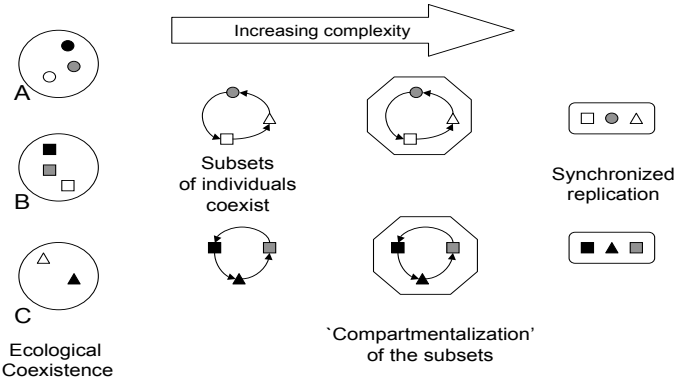


Fig. 1. Abstract model of Symbiogenesis: Complexification as a process of compartmentalization (adopted from [28]). Individuals from independent species A, B, C already under an ecological coexistence (far left) form an increasingly intimate coevolutionary partnership resulting in a 'compartment' (center) that supports the coordinated replication of symbionts (right).

different species were involved and that the process establish an intimate association over a 'significant time period.' Note also that the concept of a 'host' is quite abstract; the host may or may not be a currently existing entity.

Given this definition for the generic process of symbiogenesis – effectively establishing symbiosis as an operator rather than a state [6] (i.e., outcomes are independent of the coevolutionary interaction) – a wide range of 'resolutions' exist that provide specific examples of symbiosis in nature: **(1) Endosymbiotic:** Interactions that take place within a host potentially resulting in symbiogenesis. *Intracellular* – in which integration at the cellular level takes place. Symbiont cells enter the host, survive, reproduce and successfully appear in the host offspring e.g., as in the case of prokaryote to eukaryote transfers; *Extracellular* – symbionts establish themselves between the host cells (as opposed to within them) or within cavities of the host e.g., as in the case of the mammalian gut (host) and *E. coli* (symbiont) a relation that enables mammals to digest food. **(2) Entosymbiotic:** Represent symbiotic relationships that do not enter the host 'body' and to date lack complete histories supporting symbiogenesis [20]. Daida *et al.* separate this into two forms [6]: *Attachment Symbiosis* – the host and symbiont undergo a permanent/ semi-permanent attachment e.g., as in the case of sea anemones riding the shells of hermit crabs. As such the relationship tends to be one-to-one; *Behavioral Symbiosis* – rely on communication as the medium to establish the basis for the symbiotic association. As such, relationships can be much more flexible with hosts being served by different symbionts over time e.g., the tooth cleaning relationship between crocodiles and birds.

In addition to this, intracellular endosymbiosis can be augmented by other processes such as horizontal gene transfer (HGT) [1]. Classically, HGT was associated with the transfer of plasmids – bacterial genes not in the main chromosome; where

genetic material in the plasmid is known to confer resistance to antibiotics through the combination of high rates of plasmid exchange and a 'noisy' process of transcription [1]. The process frequently appears in bacteria, but also considered to result in the migration of plastids and mitochondria between bacteria, archaea and eukarya [33]. From the view of molecular evolution, both HGT and symbiosis imply that molecular development did not follow a tree of life, but a network of life metaphor in which there is much more interbreeding of the gene pool [33,25]. However, the underlying message is that the process of variation is that of Lamarckian inheritance augmented with mutation [26].

The above two level ontology is by no means the only scheme for distinguishing between different forms of symbiosis. Indeed, we maintain in this work that it is the relationship supporting the process of compartmentalization (Figure 1) that more effectively summarizes developments under EC. Thus, five basic categories of relationship might be identified (adapted from [24]): **(1) Spatial relationships** – what degree of physical proximity is necessary to support the identification of potential partners e.g., commensalism (a very intimate integration of partners) versus mutualism (a purely behavioral compartment in which participants maintain physical independence); **(2) Temporal relationships** – defines over what period of the participant's lifetime the compartmentalization exists e.g., whether a compartment is only established following: appropriate communication (therefore symbiosis is an occasional behavioral activity), under a periodic model of reformation and disengagement, or be permanent through the lifetime of participants; **(3) Metabolic relationships** – to what degree a third party host is necessary to provide the compartmentalization, in contrast with symbionts relying on a non-physical identification (of compartmentalization). This might raise secondary factors such as to what degree participants provide mutually beneficial food sources; **(4) Genetic relationships** – To what degree specific protein(s)/ gene(s) of a participant are transfered to others; and, **(5) Coevolutionary relationships** – symbionts need not be purely mutualistic in their interaction [26,27,6]. Indeed coevolutionary relationships could fall under the categories of amensalism, commensalism, competition, predation or mutualism.

Finally, we note that more recent observations from the field of theoretical biology have increasingly emphasized that symbiosis is associated with conferring robustness to the resulting biological entity. The resulting hypothesis of 'self extending symbiosis' refers to a process by which [18]: "evolvable robust systems continue to extend their system boundary [a.k.a compartmentalization] by incorporating foreign biological forms to enhance their adaptive capability against environmental perturbations and hence improve their survivability and reproduction potential." In short, Mendelian genetics is associated with providing the genomic architecture, whereas symbiosis extends the host through new 'layers of adaptation' [18].

3 Summary of EC Models Supporting Symbiogenesis

In the following we review examples from EC in which symbiogenesis has played a central role. Particular attention is given to the relationship supporting

compartmentalization (spatial, temporal, metabolic, genetic or coevolutionary). That is to say, it is the *combination of relations* that promotes the state of symbiosis as opposed to the *relative resolution* at which symbiosis takes place.

3.1 Learning Classifier Systems (LCS)

Initial research used the control of a 4 legged robot as the environment to consider issues such as [3]: (1) the allocation of rules from a fixed sized population to one of the four robot legs (speciation); (2) symbiosis as used to control the identification of genetic information transferred between pairs of LCS rules. Thus speciation controls the number of legs to which members of the rule population are mapped, whereas symbiosis provided an operator for pairing rules initially associated with each leg. The authors continue with this theme in later work [34]. They note that in order to promote the identification of effective serial combinations of rules, the symbiotic operator needs to focus on rules from different niches and be biased towards matching the rules that were sequentially successful. Moreover, additional controls were necessary in order to build suitable mechanisms for effective credit assignment – or temporal persistence – when using symbiosis. Once integrated, the model was able to provide favorable solutions under a suite of 'Woods' reinforcement domain problems. The focus of the earlier work was necessarily directed towards the impact of assuming different mechanisms for establishing the 'compartmentalization' of symbionts (or the *spatial* relationships of Section 2), while simultaneously providing the basis for providing solutions to a specific problem domain. Conversely, under the later work, the key factor was the use of *temporal* relationships as the mechanism for establishing stable compartments. Both approaches make use of macro operators for selecting individuals to appear in the host compartment and assume a relatively loose model of *metabolic* relation.

3.2 Symbiogenesis and Genetic Linkage Learning

The overall goal of these algorithms is to establish a mechanism for dealing with deceptive linkage/ epistasis in binary representations i.e., correlation of gene changes with fitness is highly non-linear. As such, the participants take the form of a Genetic Algorithm (GA) and most emphasis is placed on establishing relevant *metabolic* and *genetic* relationships for compartmentalization to take place. Conversely, the spatial and temporal relationships remain intimate and permanent respectively. The process involved takes the form of either reordering the genes of the host [32] or providing mechanisms for inserting different genetic information within the context of an initial host individual [9,37,35]. In the latter case, the work of Dumeur defines a structure for building solutions out of multiple symbionts in which the frequency of seeing similar values in the same gene location makes the utility of that value more probable [9]. The process for promoting symbiont membership is driven by how 'open' a host is to incorporating new symbionts. The 'weaker' a host, the more likely that it will accept

new symbionts and vice versa.[1] The GA representation utilizes a pair of values <gene location, gene value> where gene values are binary i.e., implementing a binary GA.

A different approach is taken by the 'composition' model of Watson and Pollack [37]. Again a binary GA is considered in which individuals only specify subsets of the genes. Other genes are considered 'neutral' to that individual: such neutral genes have no contribution other than to establish the alignment of the remaining genes. Fixed length individuals are assumed in order to establish gene alignment for the sharing of genetic material during symbiosis. Symbiosis is the sole mechanism by which individuals are combined to produce a child. To do so, a rule for combining non-neutral genes is established (referred to as 'composition'). In a later work this is combined with a Pareto based competitive coevolutionary model for determining whether a (symbiotic) child is retained [38,36]. Thus a child is retained if it is better than the parents, in the Pareto sense, over a random sample of training scenarios (i.e., a test for symbiogenesis). Thus, children are only accepted if they are explicitly better than the parents.

Further efforts have been made to provide a more formal structure by which composition may evolve solutions to problems with higher orders of linkage [15]. Recent results are along these lines [16] i.e., hierarchical model building . Additional refinements to the composition model have also been introduced [13]: (1) mutation for supporting population diversity (2) initial population limited to single (non-neutral) genes but allowed to incrementally increase, thus making the hierarchical gene linkage learning more explicit; and, (3) maintenance of a worst case tabu list of poorly performing genomes to bias against revisiting poor states during symbiosis. Moreover, the same process was also employed for evolving fuzzy rules under a LCS context [2]. Watson has also continued to develop the model, with a particular focus on the criteria for detecting 'good' symbiotic partners [29], dropping the requirement for children to strictly better their parents. Finally, a 'Symbiogenetic Coevolutionary' framework also concentrates on the linkage learning problem under binary GAs with symbionts having the capacity to 'inject' themselves into the host chromosome, over-writing sequences of bits [35]. Again binary deceptive problems were used to illustrate the effectiveness of the approach under a fixed length representation.

3.3 Pairwise Symbiogenesis and Coevolution of Symbiont Behaviors

Kim *et al.* develop a model for the pairwise construction of symbionts [17]. Emphasis was placed on the exchange process for mapping participants between independent symbiont and component populations i.e., *spatial* and *temporal* relationships are used to control the mapping from independent species to compartmentalization (and viceversa). However, a natural penalty for this is that there is no process for combining more than two partners in a symbiotic relation.

[1] Incidentally, the early GA model of Daida *et al.* also made use of 'switches' to indicate whether the corresponding gene of the host can be 'infected' with a symbiont of a 'type' also declared by the host [7].

The pairwise limitation also appears in the 'linear' model of Morrison and Oppacher [31]. What is particularly interesting in their linear model is that different pairwise associations are initialized to represent different *coevolutionary* relation: amensalism, commensalism, competition, predation and mutualism. Moreover, the relative 'strength' of an association can be pre-specified as a design parameter. Defining the relevant strength parameter, however, was observed to be problem dependent. Eguchi *et al.* address this by letting the association itself evolve, this time under a multi-agent context [10]. Specifically, pairs of agents are selected – 'self' and 'opponent' – as well as the children of the 'self' individual. Pairwise evaluation under a Pareto framework is then performed under each of the models of symbiotic association to establish their preferred relation. (In an earlier work the authors describe an approach based on fuzzy rules [14]).

3.4 Models with Dissimilar Representations and Multiple Populations

The evolution of neural networks provided an early example in which different representations are employed for compartment and symbiont or hierarchical Symbiotic Adaptive Neuroevolution [30]. Specifically, a 'blueprint' population in this case expresses the compartment by indexing (symbiont) neurons from an independent neuron population; thus model building is a combinatorial search over the set of symbionts i.e., a *spatial* relationship. Similarly, the symbiogenetic evolution of Genetic Programming (GP) has also been considered for 'teaming' – that is forming teams of programs which collectively evolve to provide solutions [21]. The Symbolic Bid-based (SBB) GP framework utilizes a GA to conduct a combinatorial search for effective GP symbionts; thus each GA (host) individual defines a compartmentalization. Central to this model is an explicit separation of learning when to act (the bid or a *temporal* relation) and what to do (the action) or Bid-based GP. Without this, the context under which each symbiont program operated would be lost. Results demonstrate effectiveness at problem decomposition under classification [21] and reinforcement learning domains [22]. Moreover, the symbiont population (Bid-based GP) content evolves under mutation and a variable size population model in order to support symbiogenesis in the best compartments (teams) with fitness sharing providing additional support for diversity. Finally, under a Tree structured GP context the evolution of constants using a separate GA representation/ population was considered [5]. As such this may be interpreted as symbiosis where multiple GP populations are evaluated using constants suggested by the GA population.

4 Discussion

A characterization of the form of symbiosis employed in EC is established through emphasizing the nature of relationships used to support compartmentalization. As such, *genetic* and *metabolic* relationships appear to be the norm in (binary) GAs with symbiogenesis having strong implications for solving problems with

hierarchical relationships. LCS augment *genetic* relationships with *temporal* relationships. The result is better properties for either constructing combinations of rules (LCS) or much stronger mechanisms for resolving complex gene linkage (GA), as illustrated in the case of solutions under binary deceptive or hierarchical building block style problem domains. *Spatial* and *temporal* relationships appear as a central element to the model of Kim *et al.*, whereas the other pairwise models of symbiosis emphasize the evolution of the degree of mutualism versus competition or the *coevolutionary* relationship. When multiple populations are employed with different representations – as in Neural Evolution or GP – then *spatial* and *temporal* relationships again establish the relevant model of compartmentalization for symbiogenesis to take place.

Common to symbiosis in general is the explicit support for a divide and conquer approach to evolution. EC frequently assumes sexual recombination (crossover) as the principle mechanism for making use of modularity. However, as demonstrated by the work of Watson [36], crossover requires very favorable gene orderings for addressing problems with high orders of gene linkage. Likewise, EC models making use of symbiosis require support for suitable contextual information. Models of gene alignment play a significant role in GAs supporting symbiogenesis whereas for the GP setting of SBB the concept of bidding is central to enforcing a relevant behavioral context. Moreover, diversity maintenance in the symbiont population must be explicitly addressed in order to avoid premature convergence [13,29]. Indeed, any scheme of model building through symbiosis must be augmented by suitable variation operators. This brings the discussion back to the relation between Darwinism and Symbiogenesis. It is increasingly apparent that mutation operates at many levels – micro, macro, mega [26] – with symbiosis often considered a form of mega mutation, whereas more gradualist forms of adaptation are associated with micro and macro models of mutation [26]. With this in mind Watson considered 'compositional evolution' in general as support for combining genetic material that was "semi-independently preadapted in parallel" [36]. This covers more than just symbiotic models, including specific forms of sexual recombination (implying that specific conditions for population diversity and genetic linkage exist [36]) and horizontal gene transfer (see for example 'Transgenetic Algorithms' [12]).

Finally, from the perspective of future developments, the advent of recursively applied symbiotic operators is likely. Specifically, hosts reaching symbiogenesis may themselves become candidate symbionts for the continued development of more complex individuals. This is particularly likely when the host (compartment) population make use of cooperative coevolutionary mechanisms, such as fitness sharing, to encourage diversity at the host level. The next (recursive) application of symbiosis would use (some subset of) a previously evolved host population as the candidate symbionts for building new host compartments (see for example the diversity in host/ team behaviors illustrated by [22]); thus, providing an automated process for 'layered learning.'

Acknowledgements

P. Lichodzijewski was supported in part through Killam Predoctoral and NSERC PGSD scholarships. M. Heywood was supported from an NSERC research grant.

References

1. Amabile-Cuevas, C.F., Chicurel, M.: Horizontal gene transfer. American Scientist 81, 332–341 (1993)
2. Baghshah, M.S., Shouraki, S.B., Halavati, R., Lucas, C.: Evolving fuzzy classifiers using a symbiotic approach. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1601–1607 (2007)
3. Bull, L., Fogarty, T.C.: Evolutionary computing in multi-agent environments: Speciation and symbiosis. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 12–21. Springer, Heidelberg (1996)
4. Burke, D.S., Jong, K.A.D., Grefenstette, J.J., Ramsey, C.L., Wu, A.S.: Putting more genetics into Genetic Algorithms. Evolutionary Computation 6(4), 387–410 (1998)
5. Cagnoni, S., Rivero, D., Vanneschi, L.: A purely evolutionary memetic algorithm as a first step towards symbiotic coevolution. In: Proceedings of the Congress on Evolutionary Computation, pp. 1156–1163. IEEE Press, Los Alamitos (2005)
6. Daida, J.M., Grasso, C.S., Stanhope, S.A., Ross, S.J.: Symbionticism and complex adaptive systems I: Implications of having symbiosis occur in nature. In: Proceedings of the Annual Conference on Evolutionary Programming, pp. 177–186. MIT Press, Cambridge (1996)
7. Daida, J.M., Ross, S.J., Hannan, B.C.: Biological symbiosis as a metaphor for computational hybridization. In: Proceedings of the International Conference on Genetic Algorithms, pp. 328–335. Morgan Kaufmann, San Francisco (1995)
8. de Bary, H.A.: Die Erscheinung der Symbiose. Votrag, gehalten auf der Versammlung Deutscher Naturforscher und Aerzte zu Cassel (1879)
9. Dumeur, R.: Evolution through cooperation: The symbiotic algorithm. In: Alliot, J.-M., Ronald, E., Lutton, E., Schoenauer, M., Snyers, D. (eds.) AE 1995. LNCS, vol. 1063, pp. 145–158. Springer, Heidelberg (1996)
10. Eguchi, T., Hirasawa, K., Hu, J., Ota, N.: A study of evolutionary multiagent models based on symbiosis. IEEE Transactions of Systems, Man, and Cybernetics–Part B 36(1), 179–193 (2006)
11. Fogel, D.B. (ed.): Evolutionary Computation: The Fossil Record. IEEE Press, Los Alamitos (1998)
12. Goldbarg, E.F.G., Goldbarg, M.C., Bagi, L.B.: Transgenetic algorithm: A new evolutionary perspective for heuristics design. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 2701–2708 (2007)
13. Halavati, R., Shouraki, S.B., Heravi, M.J., Jashmi, B.J.: Symbiotic evolutionary algorithm: A general purpose optimization approach. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 4538–4545 (2007)
14. Hirasawa, K., Ishikawa, Y., Hu, J., Murata, J., Mao, J.: Genetic symbiosis algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1377–1384 (2000)
15. de Jong, E., Thierens, D., Watson, R.A.: Hierarchical genetic algorithms. 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. 232–241. Springer, Heidelberg (2004)

16. de Jong, E., Watson, R.A., Thierens, D.: On the complexity of hierarchical problem solving. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1201–1208. ACM Press, New York (2005)
17. Kim, J.Y., Kim, Y., Kim, Y.K.: An endosymbiotic evolutionary algorithm for optimization. Applied Intelligence 15, 117–130 (2001)
18. Kitano, H., Oda, K.: Self-extending symbiosis: A mechanism for increasing robustness through evolution. Biological Theory 1(1), 61–66 (2005)
19. Kutschera, U.: Symbiogenesis, natural selection, and the dynamic earth. Theory in Biosciences 128, 191–203 (2009)
20. Kutschera, U., Niklas, K.J.: Endosymbiosis, cell evolution, and speciation. Theory in Biosciences 124, 1–24 (2005)
21. Lichodzijewski, P., Heywood, M.I.: Managing team-based problem solving with symbiotic bid-based Genetic Programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 363–370 (2008)
22. Lichodzijewski, P., Heywood, M.I.: Binary versus real-valued reward functions under coevolutionary reinforcement learning. In: Proceedings of the International Conference on Artificial Evolution (2009), https://lsiit.u-strasbg.fr/ea09
23. Margulis, L.: Symbiosis and evolution. Scientific American 225(2), 48–57 (1971)
24. Margulis, L.: Symbiogenesis and Symbionticism, ch. 1, pp. 1–14 (1991) in ([26])
25. Margulis, L.: Genome acquisition in horizontal gene transfer: Symbiogenesis and macromolecular sequence analysis. In: Gogarten, M.B., et al. (eds.) Horizontal Gene Transfer: Genomes in Flux, ch. 10, pp. 181–191. Springer, Heidelberg (2009)
26. Margulis, L., Fester, R. (eds.): Symbiosis as a Source of Evolutionary Innovation. MIT Press, Cambridge (1991)
27. Margulis, L., Sagan, D.: Acquiring Genomes. Basic Books (2002)
28. Maynard Smith, J.: A Darwinian View of Symbiosis, ch. 3, pp. 26–39 (1991), in ([26])
29. Mills, R., Watson, R.A.: Symbiosis, synergy and modularity: Introducing the reciprocal synergy symbiosis algorithm. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 1192–1201. Springer, Heidelberg (2007)
30. Moriarty, D.E., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution. Evolutionary Computation 5(4), 373–399 (1998)
31. Morrison, J., Oppacher, F.: A general model of coevolution for genetic algorithms. In: Proceedings of Artificial Neural Networks and Genetic Algorithms (1999)
32. Paredis, J.: The symbiotic evolution of solutions and their representations. In: Proceedings of the International Conference on Genetic Algorithms, pp. 359–365. Morgan-Kaufmann, San Francisco (1995)
33. Smets, B.F., Barkay, T.: Horizontal gene transfer: Perspectives at a crossroads of scientific disciplines. Nature Reviews Microbiology 3, 675–678 (2005)
34. Tomlinson, A., Bull, L.: Symbiogenesis in learning classifier systems. Artificial Life 7, 33–61 (2001)
35. Wallin, D., Ryan, C., Azad, R.M.A.: Symbiogenetic coevolution. In: Proceedings of the Congress on Evolutionary Computation, pp. 1613–1620. IEEE Press, Los Alamitos (2005)
36. Watson, R.A.: Compositional Evolution: The impact of sex, symbiosis and modularity on the gradualist framework of evolution. MIT Press, Cambridge (2006)
37. Watson, R.A., Pollack, J.B.: How symbiosis can guide evolution. In: European Conference on Artificial Life, pp. 29–38. Springer, Heidelberg (1999)
38. Watson, R.A., Pollack, J.B.: A computational model of symbiotic composition in evolutionary transitions. BioSystems 69, 187–209 (2003)

Co-evolution of Optimal Agents for the Alternating Offers Bargaining Game

Arjun Chandra, Pietro Simone Oliveto, and Xin Yao

The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, UK
{a.chandra,p.s.oliveto,x.yao}@cs.bham.ac.uk

Abstract. Bargaining, as an instance of sequential games, is a widely studied problem in game theory, experimental and computational economics. We consider the problem of evolving computational agents with optimal (Subgame Perfect Equilibrium) strategies for the Alternating Offers Bargaining Game. Previous work co-evolving agents for this problem has argued that it is not possible to achieve optimal agents at the end of the co-evolutionary process due to the *myopic* properties of the evolutionary agents. Emphasising the notion of a *co-evolutionary solution concept*, we show that this conclusion is mis-leading and present a co-evolutionary algorithm that evolves optimal strategies for the bargaining game with one round. We conclude by explaining why, using previous evaluation procedures and strategy representations, the algorithm is not able to converge to optimal strategies for games with more rounds.

1 Introduction

Co-evolution is a process of mutual adaptation, towards increasingly adaptive behaviours, that occurs amongst a set of agents interacting strategically (revealing a reward structure) in some domain. It has been used for both the evolution of game playing strategies [8] in order to achieve a certain type of agent behaviour, and at the same time using games as a test bed to understand the co-evolutionary process properly [4]. However, the reported successes of co-evolutionary applications are counter balanced by many failures [5]. These failures are often attributed to co-evolutionary properties such as *cyclic dynamic, mediocre stable-state, collusion, forgetting* etc [5]. It has been shown that often these pathologies imply a lack of rigour in the definition of the *solution concept*[1] used for the co-evolutionary process [5]. Recently, generalisation has been proposed and studied rigorously as an alternative to measuring strategy performance quantitatively and unambiguously [3].

Sequential games, for which we have a closed form solution, specifically, a single selected Nash Equilibrium i.e. Subgame Perfect Equilibrium (SPE) [11],

[1] In co-evolutionary terms, a solution concept is defined by defining two search problems [5]: (1) searching for the desired solution i.e. SPE, and (2) searching for a direction in terms of parts of the landscape/opponents that guide the search for the desired solution.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 61–70, 2010.

are useful for building, validating, and refining bottom-up approaches (like co-evolution) to the equilibrium selection process. They provide us with a theoretically calculated optimum, and hence a solution concept, to aim for. This gives support to the designed co-evolutionary algorithm to be applicable for equilibrium selection in game settings where the theoretical equilibrium is intractable. This is one of the fundamental ideas behind the field of computational economics [9].

We consider the problem of co-evolving optimal agents for the *alternating offers bargaining game* [10]. Previous work [2] using co-evolution for finding the theoretical equilibrium (SPE) for the game suggested that it was not possible to achieve optimal agents at the end of the co-evolutionary process, hence, that the algorithm did not converge. This was attributed to the agents being *myopic* or *boundedly rational*, in that the agents had no memory to remember past interactions, no explicit rationality principles to use, and for more realistic game settings with a stochastic element defining when the game ends, they were supposedly unable to reason backwards from the deadline. While these issues may be interesting, they do not directly answer why co-evolution, in this case, does not lead to optimality and convergence. Since a given setting of the bargaining game has only one SPE, it is worth investigating whether a co-evolutionary algorithm can evolve the agents' strategies towards optimality. In this paper we further analyse the co-evolutionary algorithm used in [2,7] to understand why optimal agents cannot be co-evolved and whether there are better co-evolutionary approaches that can be used in this case. The goal here is to identify general issues that may influence co-evolution, not just for a particular game considered.

The rest of this paper is structured as follows. In Section 2, we introduce the alternating offers bargaining game and the co-evolutionary setting used in [7]. In Section 3, we analyse the previously used methodology and explain why the conclusions were mis-leading. In Section 4, we present a co-evolutionary algorithm that evolves strategies converging to theoretical optimality for games with a single round. We then explain why the same results cannot be obtained for games with more than one round using previous evaluation procedures and strategy representations, influencing the general applicability of these algorithms to sequential games, hence why the implemented solution concept has to be considerably modified. We conclude in Section 5 by discussing ideas for future work directed towards the obtainment of a fully convergent co-evolutionary algorithm for the alternating offers bargaining problem with multiple rounds.

2 Preliminaries

2.1 Alternating Offers Multiple Issue Bargaining Game

We consider the multiple issue, finite horizon (finite number of rounds) version of Rubinstein's [10] alternating offers bargaining game. There are two agents, 'Player 1' and 'Player 2', and Player 1 always starts the game. A round entails each agent (as an *offerer*), in turn, proposing an offer (a division of the surplus on m issues, expressed as a vector $\boldsymbol{o}$) to the other agent (i.e. the *responder*) and

the responder deciding whether or not to accept this offer by matching the utility of the offer against a threshold. There are a maximum of n rounds. At the end of each round, the game prematurely breaks down with a probability $1-p$. The probability p (of the game proceeding to the next round) reflects the players' uncertainty about the deadline in real world bargaining interactions. The game ends in a disagreement if there is a breakdown or if the responder rejects the offer in the last round. In either case both agents receive nothing (i.e. the utility is zero). The game ends in an agreement if the responder accepts the offer in any of the rounds. In this case, the agents receive a positive payoff decided by their respective utility functions.

As in [7], we assume, without loss of generality, that the total bargaining surplus available for each issue is *unity*. For either agent, the utility of an offer $\boldsymbol{o}$ is the dot product $\boldsymbol{w_j}.\boldsymbol{o} = \sum_{i=1}^{m} w_j^i o^i$, where $\boldsymbol{w_j}$ is a vector containing agent j's weights for each issue and $\sum_{i=1}^{m} w_j^i = 1$. In the rest of the paper we fix the number of issues (m) to 2, $\boldsymbol{w_1} = (0.7, 0.3)$ and $\boldsymbol{w_2} = (0.3, 0.7)$, since the majority of the results reported in [7] consider these values. Thus, a game is instantiated by defining a specific value for p and n. For each combination of p and n, there is a unique SPE solution that can be calculated using backward induction (refer to [6]).

As an example, we consider the game with no breakdown i.e. $p = 1$. In this case, the game theoretic solution (SPE) is the obvious one, where the first agent gets everything on all issues (payoff of 1) and the second gets nothing (payoff of 0) if the number of rounds n is odd. On the other hand, if the number of rounds n is even, then the first agent gets nothing and the second gets everything on all issues. In other terms, if the number of rounds is fixed, then the agent that turns out to be the responder in the last round should accept any offer since the alternative is disagreeing, hence receiving no utility at all (it does not gain anything from disagreeing).

2.2 The Co-evolutionary Algorithm

We apply the same co-evolutionary self adaptive $(\mu+\lambda)$-Evolution Strategy (ES) as that used in [7]. Two populations (of size μ) are considered, one for each type of agent (Player 1 or Player 2). Each individual in a population plays a game with every individual in the other population, and the fitness is evaluated as the average utility that the individual gets from all these games. At each generation, λ individuals are selected from each population uniformly at random (independently) and then mutated to form the two offspring populations (of size λ), one for each agent type. The individuals in the offspring populations are evaluated by playing against every individual in the opponent parental population. The fitness of each individual is the average utility obtained by playing against these opponents. The μ fittest individuals are selected from the respective pools of $\mu+\lambda$ individuals of both types for the next generation. Fig. 1 shows a schematic diagram of this process. We replicate the same algorithmic parameters used

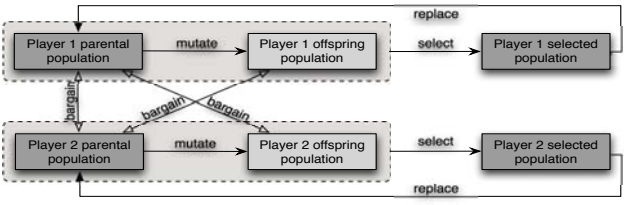


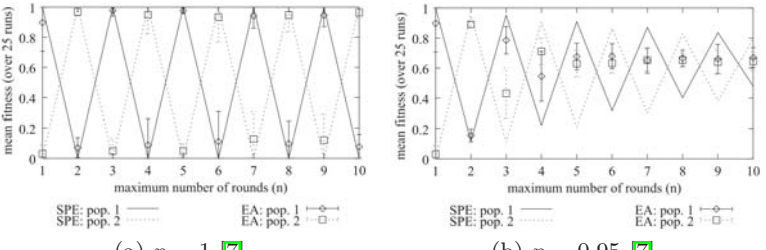
Fig. 1. Co-evolutionary $(\mu + \lambda)$-ES

in [2] i.e. $\mu = \lambda = 25$, initial $\sigma_i = 0.1$ and a lower bound on σ_i s.t. $\sigma_i \geq 0.025$. Also, the strategy representation is the same as in [7]. The agent strategy specifies the offers $\boldsymbol{o_j}(r)$ and thresholds $t_j(r)$ for each round r in the game for agents $j \in \{1, 2\}$. At the start of each run, the offers and thresholds are initialised by sampling random numbers in the interval $[0.0, 1.0]$ from a uniform distribution.

3 Analysis of the Previous Methodology

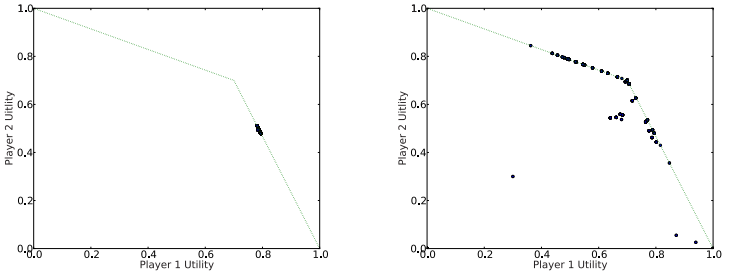
Fig. 2 shows the mismatches between evolutionary and SPE results for bargaining games with $p = 1$ and $p = 0.95$, as reported in [7]. Lines join SPE solutions to guide the eye. It can be seen that the mean fitness of the populations over 25 runs is somewhat close to the SPE values but does not converge to them. This holds for any p or any n. The reasons are attributed to the agents being *myopic* [7]. Since this myopia is reported for all values of p and n, it sounds safe to look into the most simple setting of the game, that of $p = 1$, in our following analyses.

We re-implement the methodology from [7]. When running the algorithm we see that, independently from the values of p and n, the populations indeed evolve towards the pareto-efficient frontier (Fig. 3). And, as reported in [7], the agents keep exploring the search space as a "moving cloud" of agreements along the frontier, instead of evolving towards the equilibrium. Moreover, we see that, the populations scatter in the long term, only to regroup, move along the frontier and scatter again. Fig. 3 depicts the described "breakdowns". This should not



(a) $p = 1$ [7]. (b) $p = 0.95$ [7].

Fig. 2. Comparing evolutionary results with SPE results for $p = 1$ and $p = 0.95$ [7]



(a) After 3200 generations. (b) After 3700 generations.

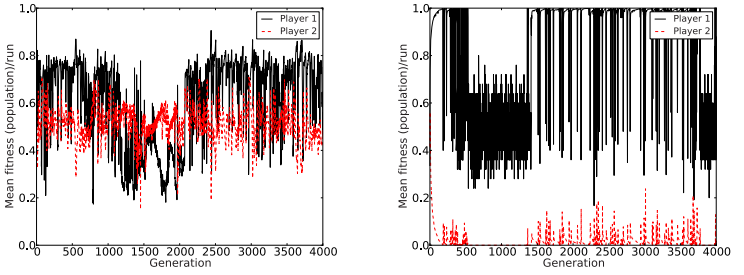
Fig. 3. Outcomes of games played (agreements reached) by both parental populations for $p = 0.7$ and $n = 10$ from a typical run. $SPE = (0.769, 0.538)$. The dotted line is the pareto-efficient frontier.

happen if our solution concept is that of finding the equilibrium for the game, especially with elitist selection.

In Fig. 4 we plot the mean fitness of the populations in a typical run for $p = 0.7$, $n = 10$ (Fig. 4(a)) and $n = 1$ (Fig. 4(b)) respectively, $p = 1$ by default in the latter. In the first case we can see that the mean fitness of both populations just goes up and down. The figure suggests that the strategies are not evolving towards any direction. The wrong co-evolutionary solution concept might have thus been applied. From Fig. 4(b) we can understand a bit more. Since for the simple game with $n = 1$, the optimal solution is 1.0 for Player 1 and 0.0 for Player 2 (as discussed in Section 2.1), the fitness trajectories in Fig. 4(b) are clearer. We see that the mean fitnesses do head towards the optimal values and then collapse repeatedly. Although the causes for this divergence still remain unclear, we see that the co-evolutionary methodology does not converge even for this simpler game. In the following we continue studying the game where $p = 1$ and simplify the experimental setup in order to gain insight into the co-evolutionary dynamics, specifically convergence.

In particular, we remove the influence of a population by considering only 1 individual per agent type in the co-evolutionary setup (i.e. we get the (1+1)-ES[2]). The idea is to study the causes of the breakdowns by looking at the individual strategies. All other parameters remain the same as in the original population based algorithm. Fig. 5(a) shows 500 generations of a typical run. We see that (as also noted for the single issue case in [2], albeit, with a population), the individuals actually reach the SPE of $(1.0, 0.0)$ for odd n and $(0.0, 1.0)$ for even n. This however should not happen. It is theoretically impossible to reach the optimum $\boldsymbol{x}^*$ of a continuos optimisation problem in finite time, but only a point $\boldsymbol{x}$ such that $|\boldsymbol{x} - \boldsymbol{x}^*| \leq \epsilon$ for any constant $\epsilon > 0$ [1]. The ES should

[2] Note that in a (1+1)-ES, the offspring replaces the parent only if its fitness is greater than or equal to that of the parent (i.e. if the parent and offspring have the same fitness, selection is not performed uniformly at random).



(a) $p = 0.7$, $n = 10$. (b) $n = 1$.

Fig. 4. Mean fitness of the populations in a typical run

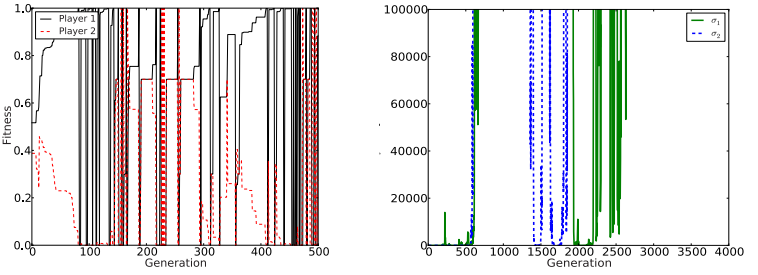
converge towards the optimum by gradually decreasing the step size $\boldsymbol{\sigma}$ and only reach the exact optimal point at infinity. Hence, there seems to be a problem with the algorithm implementation.

It turns out that the mutation operator sets any mutated value larger than unity (or smaller than zero) to unity (respectively zero) [7], instead of considering solutions outside the search space as infeasible. Although this pushes the agents' strategies to optimality (i.e. for $p = 1$), it turns out to be destructive. Once the agents reach the SPE values exactly, the responder becomes indifferent between *getting nothing* and *disagreeing* with the opponent. There is no selection pressure for the responder to keep a threshold of 0.0 (while there would have been for $0+\epsilon$ for any positive ϵ). Hence the responder ends up accepting a random threshold in the following generation with a high probability (in fact, 1) resulting in a disagreement (i.e. the fitness goes down to 0). The new threshold would not be *too* bad if the mutation step size were decreasing in time as it should. However, the step size $\boldsymbol{\sigma}$ ends up evolving in the opposite way. The higher the $\boldsymbol{\sigma}$, the higher is the probability that an infeasible solution is created which in turn is transformed into the optimal one. In Fig. 5(b) we plot the $\boldsymbol{\sigma}$ values for player 1 in a typical run. The same phenomenon happens for player 2. Where the values of σ_i's are not visible, they are higher than the plotted extreme.

4 An Improved Methodology

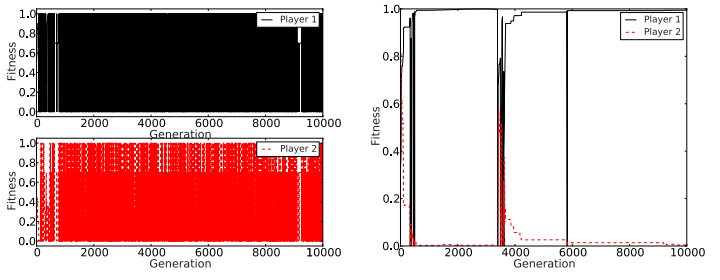
We modify the algorithm such that infeasible solutions (i.e. solutions outside the search space) have worse fitness than any feasible search space point (i.e. -1.0), hence are never accepted.

In Fig. 6, the worst run of the corrected (1+1)-ES is compared with a typical run of the original version. It clearly shows the difference this simple correction to the original approach makes. Now the strategies evolve towards optimality for long periods of time. However, breakdowns still occur (although far less frequently) and become more severe as the number of rounds n increases. We will discuss the reasons in the next section by looking, once again, at the $n = 1$ case.



(a) A typical run. (b) σ_i for Player 1.

Fig. 5. A typical run of the (1+1)-ES using the mutation procedure in [7] for $n = 1$

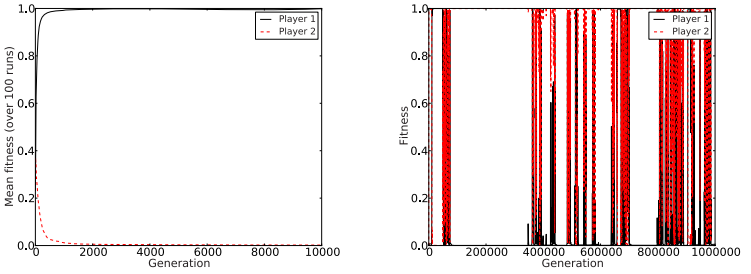


(a) A typical run. (b) Worst (most divergent) of 25 runs.

Fig. 6. Runs for the (1+1)-ES using (a) the mutation procedure in [7] (separate plots for agent types to make the severity of fluctuations unambiguous) and (b) the corrected version, for $n = 1$

4.1 Selection of Incompatible Opponents and Local Evaluation

We call a pair of strategies as *incompatible* if they disagree when playing with each other. A closer look at the evolved strategies reveals that the simultaneous selection of *incompatible* offspring as the new parents leads to a disagreement, and hence, a collapse. To illustrate this, we call P_1 and P_2 the two parents (one for each agent type) at any given time during the co-evolutionary process, and the offspring C_1 and C_2 respectively. If (P_1, P_2) are compatible, (P_1, C_2) are compatible, and (C_1, P_2) are compatible, it does not necessarily follow that (C_1, C_2) will be compatible. We argue that this was going un-noticed in the original work [7] and being attributed to the *myopic* properties of the agents. Note that an agent in equilibrium can be myopic in the sense that it may only know how to play against another agent in equilibrium, which is when the issue of myopia should be scrutinised. This is not the case here. Having not played a game before, if the offspring are incompatible and selected to be in the next generation, then they will inevitably disagree. This problem (i.e. *selection of incompatible*



(a) $n = 1$. (b) $n = 2$.

Fig. 7. Modified (1+1)-ES for (a) $n = 1$ (mean across 100 runs) and (b) $n = 2$ (a typical diverging run)

opponents) does not depend on *myopic* properties of agent strategies, but on an unusual implementation of the co-evolutionary solution concept.

In the following, we modify the methodology by allowing the offspring to play games against each other before selection. If this game results in a disagreement, the offspring are not accepted, otherwise, the co-evolutionary process carries on as in the previous methodology. This, by definition, eliminates the problem of *selection of incompatible opponents* altogether. Fig. 7(a) shows the mean fitness across 100 runs of the modified (1+1)-ES. No divergence was seen in any of the runs (we also allowed the algorithm to run for a million generations, without seeing one breakdown). Since there is no reason for breakdown occurring if we add populations, the algorithm will now work also for $\mu > 1$ and $\lambda > 1$. However, the modifications we have applied do not imply convergence for $n > 1$. In fact, already for $n = 2$, we observe some fitness breakdowns again. Figure 7(b) shows a run of one million generations for $n = 2$.

Zooming into Fig. 7(b), to the generations where there is a sudden change from the convergent trajectory (generations starting from 374400 for instance), and investigating the agent strategies in these generations, reveals the cause for these breakdowns. Fig. 8 takes the two agent (P_1 and P_2) strategies apart and shows what each agent receives and wants in terms of utilities in both rounds (since $n = 2$). It can be seen that the *evaluation of a strategy* is *local* with respect to the rounds. Since the fitness of an individual is computed whenever there is an agreement, which can happen in any round, the strategies only consider the agreement round to evaluate themselves (i.e. it is the offers and thresholds of the agreement round that determine the strategies' payoffs). In Fig. 8, the agreements happen in round 2 to begin with (when the outcomes are near SPE). There is no real selection pressure for modifying round 1 offers and thresholds, so the strategy chunks corresponding to this round vary randomly and at some point are such that P_1 offers more than what P_2 needs. Consequently, the agreement happens in round 1. Now, P_1 can take advantage of the fact that it is in the driver's seat (as the utility of both agents is decided by the offer it makes, when this offer is accepted, which it is), while P_2 has lost its selection pressure

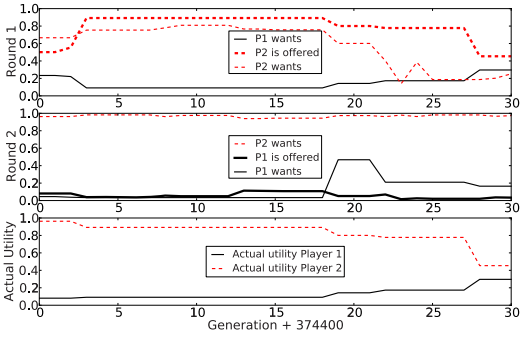


Fig. 8. Snapshot of the modified (1+1)-ES for $n = 2$. Strategy evaluation is "local" with respect to the rounds.

towards higher fitness (mismatch between its threshold and P_1's offer leading to P_2 getting what it is given). Meanwhile, the selection pressures for offers and thresholds in round 2 vary randomly since the players agree in round 1 and never reach round 2. By modifying the fitness evaluation procedure such that it credits the whole strategy instead of just the part corresponding to the agreement round, we may solve the problem. Alternatively, had the representation not been treating each round as being separate from each other, we may solve the problem too. These are our immediate next steps.

Note that co-evolutionary approaches to games such as bargaining influence a general class of games i.e. sequential games. All these games specify the SPE as the optimal. Using similar types of agent interactions, selection and evaluation procedures, and representations, as scrutinised in this paper, can be causes for divergence in these games. Thus, a thorough investigation and definition of the co-evolutionary solution concept, as initiated here for bargaining games, is in order and should generally apply to sequential games. One potential direction is to adapt the theoretical work in [3] for our work here.

5 Conclusion

We presented a deeper analysis of an existing co-evolutionary approach to evolving optimal agents for multi-issue bargaining games. Three weaknesses were discovered viz. the mutation procedure in the way it dealt with infeasible solutions, the selection strategy in co-evolution in the way incompatible offspring were handled, and the evaluation or representational issue, which, independently may be responsible for the local evaluation of individuals. We were able to co-evolve optimal agents that were previously impossible to obtain by addressing the first two issues above. However, more work needs to be done to rectify the evaluation/representational issue in order to obtain optimal agents for games with more rounds. As immediate next steps, we intend to make agent behaviours in

different rounds dependent in some fashion, independently at the evaluation level and at the representational level, to further understand and remove the causes for divergence from optimality altogether. This should give us a systematically designed solution concept. We then intend to test the algorithm on more complex bargaining games, starting with $p < 1$, moving on to games involving a greater number of issues, and games where the SPE are either hard to compute by hand or even intractable, following on to other sequential games. It is important to note that, without a well defined solution concept, results of co-evolutionary approaches to problems like the one considered in this paper (a common practice in computational economics), may be mis-leading.

References

1. Beyer, H.: The theory of evolution strategies. Springer, New York (2001)
2. van Bragt, D.D.B., Gerding, E.H., La Poutré, J.A.: Equilibrium selection in alternating-offers bargaining models - the evolutionary computing approach. The Electronic Journal of Evolutionary Modeling and Economic Dynamics (2002)
3. Chong, S.Y., Tino, P., Yao, X.: Measuring generalization performance in coevolutionary learning. IEEE Transactions on Evolutionary Computation 12(4), 479–505 (2008)
4. Darwen, P.J.: Co-evolutionary learning by automatic modularisation with speciation. Ph.D. thesis, University of New South Wales (1996)
5. Ficici, S.G.: Solution concepts in coevolutionary algorithms. Ph.D. thesis, Brandeis University (2004)
6. Gerding, E., van Bragt, D.D.B., La Poutré, J.A.: Multi-issue negotiation processes by evolutionary simulation: validation and social extensions. Tech. Rep. SEN-R0024, CWI, Amsterdam, The Netherlands (2000)
7. Gerding, E., van Bragt, D.D.B., La Poutré, J.A.: Multi-issue negotiation processes by evolutionary simulation, validation and social extensions. Computational Economics 22(1), 39–63 (2003)
8. Jin, N.: Constraint-based co-evolutionary genetic programming for bargaining problems. Ph.D. thesis, University of Essex (2007)
9. Roth, A.: The economist as engineer: game theory, experimentation, and computation as tools for design economics. Econometrica 70(4), 1341–1378 (2002)
10. Rubinstein, A.: Perfect equilibrium in a bargaining model. Econometrica 50(1), 97–109 (1982)
11. Selten, R.: Re-examination of the perfectness concept for finite points in extensive games. International Journal of Game Theory 4, 25–55 (1975)

Fuzzy Nash-Pareto Equilibrium: Concepts and Evolutionary Detection

Dumitru Dumitrescu, Rodica Ioana Lung, Tudor Dan Mihoc, and Reka Nagy

Babeş Bolyai University, Cluj Napoca, Romania

Abstract. Standard game theory relies on the assumption that players are rational agents that try to maximize their payoff. Experiments with human players indicate that Nash equilibrium is seldom played. The goal of proposed approach is to explore more nuance equilibria by allowing a player to be biased towards different equilibria in a fuzzy manner. Several classes of equilibria (Nash, Pareto, Nash-Pareto) are defined by using appropriate generative relations. An evolutionary technique for detecting fuzzy equilibria is considered. Experimental results on Cournot' duopoly game illustrate evolutionary detection of proposed fuzzy equilibria.

1 Introduction

The concept of fuzzy rationality of the players in non cooperative games is introduced. Hopefully the proposed concept may capture the intrinsic fuzziness of human decision making process.

Definition 1. *A finite strategic game is defined as a system* $\Gamma = ((N, S_i, u_i), i = 1, n)$*, where:*

- N *represents a set of* n *players,* $N = \{1, ..., n\}$*;*
- *for each player* $i \in N$*,* S_i *represents the set of actions available to her,* $S_i = \{s_{i_1}, s_{i_2}, ..., s_{i_m}\}$*;* $S = S_1 \times S_2 \times ... \times S_n$ *is the set of all possible strategies (situations of the game);*
- *for each player* $i \in N$*,* $u_i : S \to R$ *represents the payoff function.*

Denote by (s_{i_j}, s^*_{-i}) the strategy profile obtained from s^* by replacing the strategy of player i with s_{i_j} i.e.

$$(s_{i_j}, s^*_{-i}) = (s^*_1, s^*_2, ..., s^*_{i-1}, s_{i_j}, s^*_{i+1}, ..., s^*_n).$$

A strategy is a Nash equilibrium [8,7,1] if each player has no incentive to unilaterally deviate i.e. it can not improve the payoff by modifying its strategy while the others do not modify theirs. More formal, the strategy s^* is a Nash equilibrium if and only if the inequality

$$u_i(s_i, s^*_{-i}) - u_i(s^*) \leq 0, \ \forall s_i \in S_i, \forall i \in N$$

holds.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 71–79, 2010.

2 Fuzzy Nash/Pareto-Biased Players

Each concept of equilibrium may be associated with a rationality type. We consider games with players having several rationality types [4]. Each player can be more or less biased towards a certain rationality. This bias may be expressed by a fuzzy membership degree. A player may have for instance the membership degree 0.7 to Nash and the membership 0.3 to Pareto. We may also say that the player has a Nash rationality defect of 0.3.

Let us consider a fuzzy set A_N on the player set N i.e.

$$A_N : N \rightarrow [0, 1]$$

$A_N(i)$ expresses the membership degree of the player i to the class of Nash-biased players. Therefore A_N is the class of Nash-biased players.

Similar a fuzzy set $A_P : N \rightarrow [0, 1]$ may describe the fuzzy class of Pareto-biased players.

The aim is to define several concepts of fuzzy equilibrium. These concepts are completely different from those considered in the framework of fuzzy games [10].

An evolutionary technique for detecting fuzzy equilibria in non cooperative games is proposed in section 6.1.

3 Fuzzy Nash Equilibrium

A fuzzy Nash equilibrium concept using an appropriate generative relation that includes the membership degrees to fuzzy class Nash-biased players is defined in this section. The new concept is a natural generalization of Nash equilibrium characterization by generative relations [4].

3.1 Generative Relation for Nash Equilibrium

Let x and y be two pure strategies and $k(y, x)$ denotes the number of players which benefit by deviating from y towards x:

$$k(y, x) = card\{i | u_i(x, y_{-i}) > u_i(y)\}.$$

$k(y, x)$ is a relative quality measure of y and x – with respect to the Nash equilibrium.

Strategy y is better than strategy x with respect to Nash equilibrium, and we write $y \prec x$, if and only if

$$k(y, x) < k(x, y)$$

We may consider $\prec$ as a generative relation of Nash equilibrium, i.e. the set of nondominated strategies with respect to $\prec$ equals the Nash equilibrium [5].

3.2 Generative Relation for Fuzzy Nash Equilibrium

Our aim is now to define a generative relation for fuzzy Nash equilibrium. In this respect the relative quality measure of two strategies has to involve the fuzzy membership degrees.

Let us consider the threshold function:

$$t(a) = \begin{cases} 1, & \text{if } a > 0, \\ 0, & \text{otherwise} \end{cases}$$

The fuzzy version of the quality measure $k(y, x)$ is denoted by $E_N(y, x)$ and may be defined as

$$E_N(y, x) = \sum_{i=1}^{n} A_N(i)t(u_i(x, y_{-i}) - u_i(y)).$$

$E_N(y, x)$ expresses the relative quality of the strategies y an x with respect to the fuzzy class of Nash-biased players.

Let us consider the sharp (classical) relation $\prec_N$ defined as:
$y \prec_N x$ if and only if the inequality

$$E_N(y, x) < E_N(x, y)$$

holds.

Fuzzy Nash equilibrium is defined as the set of nondominated strategies with respect to the relation $\prec_N$.

Remark 1. By definition $\prec_N$ is the generative relation of the fuzzy Nash equilibrium.

Remark 2. Fuzzy generative relations may also be defined in a similar way.

4 Fuzzy Pareto Equilibrium

Let us define the quantity $P(y, x)$ as the number of players having a better payoff for x than for y:

$$P(y, x) = card\{i | u_i(y) < u_i(x)\}.$$

Consider the relation $\ll$ defined as $y \ll x$ if and only if

$$P(y, x) < P(x, y).$$

We may admit that the relation $\ll$ expresses a certain type of Pareto rationality i.e. each player is trying to maximize its payoff irrespective to the other players options. Otherwise stated the nondominated strategies with respect to the relation $\ll$ represents a variety of Pareto equilibrium.

The fuzzy version of $P(y, x)$ may be defined as

$$E_P(y, x) = \sum_{i=1}^{n} A_P(i)t(u_i(x) - u_i(y))$$

where A_P is the fuzzy set of the Pareto-biased players.

Consider now the relation $\ll_P$ defined as $y \ll_P x$ if and only if the inequality

$$E_P(y, x) < E_P(x, y)$$

holds.

Relation $\ll_P$ can be considered a generative relation of fuzzy Pareto equilibrium. Our aim is to investigate this type of equilibrium and combine it with fuzzy Nash equilibrium.

5 Fuzzy Nash-Pareto Equilibrium

Let us consider a game involving both Nash and Pareto-biased players. It is natural to assume that $\{A_N, A_P\}$ represents a fuzzy partition of the player set.

Therefore the condition

$$A_N(i) + A_P(i) = 1$$

holds for each player i.

A fuzzy version of the sharp Nash-Pareto equilibrium introduced in [4] can be considered. The relative quality measure of the strategies y and x with respect to Nash-Pareto (fuzzy) rationality may be defined as

$$E(y, x) = E_N(y, x) + E_P(y, x).$$

Therefore

$$E(y, x) = \sum_{i=1}^{n} A_N(i) t(u_i(x, y_{-i}) - u_i(y)) + \sum_{i=1}^{n} A_P(i) t(u_i(x) - u_i(y)).$$

Using the relative quality measure E we can compare two strategy profiles. Let us introduce the relation $\prec_{fNP}$ defined as $y \prec_{fNP} x$ if and only if the strict inequality

$$E(y, x) < E(x, y)$$

holds.

We define a new equilibrium called *fuzzy Nash-Pareto equilibrium* as the set of non-dominated strategies with respect to the relation $\prec_{fNP}$.

6 Numerical Experiments

Several numerical experiments have been conducted in order to illustrate evolutionary detection of the proposed equilibria concepts.

6.1 Evolutionary Computing of Fuzzy Equilibria

Fuzzy equilibria can be detected by evolving a population of strategy profiles based on generative relations [4]. For a certain equilibrium the corresponding

generative relation allows the comparison of two strategies. This comparison may guide the search of an evolutionary algorithm towards the game equilibrium.

Nondomination (with respect to a generative relation) is considered for fitness assignment purposes. Evolutionary Multiobjective Optimization Algorithms [3,2] are efficient tools for evolving strategies based on a nondomination relation.

The state of the art NSGA2 [2] has been considered to illustrate how generative relations can be used for evolutionary detection of the proposed fuzzy equilibria.

A population of 100 strategies has been evolved using a rank based fitness assignment technique. In all experiments the process converges in less than 30 generations.

6.2 Cournot Model

The following Cournot model (see [9]) is considered for numerical experiments.

A single good is produced by n firms. The cost to firm i of producing q_i units of the good is $C_i(q_i)$, where C_i is an increasing function (more output is more costly to produce). All the output is sold at a single price, determined by the demand for the good and the firms total output. If the firms total output is Q, than the market price is $P(Q)$. Market price can be expressed as $P(Q) = a - Q$ if $Q \leq a$ and 0 otherwise. If the output of each firm is q_i, then the price is $P(q_1 + q_2 + ... + q_n)$ and the firm i' revenue is $q_i P(q_1 + q_2 + ... + q_n)$. The payoff function for the player i is:

$$\pi_i(q_1, ..., q_n) = q_i P(q_1 + ... + q_n) - C_i(q_i)$$

Suppose there are two firms, each firm cost function is $C_i(q_i) = cq_i$ for all q_i. The profit of the firm i is:

$$\begin{aligned} \pi_i(q_i, q_j) &= q_i P(Q) - C_i(q_i) \\ &= q_i \left[a - (q_i + q_j) - c\right]. \end{aligned}$$

In the following experiments we consider $a = 24$ and $c = 9$.

In all the figures the players' payoffs detected for different equilibrium approximations are depicted.

The payoffs associated with the standard (sharp) Nash and Pareto equilibria, detected by the evolutionary technique [4,5] are depicted in Figure 1.

6.3 Fuzzy Nash Equilibrium Detection

Consider a situation where the players are biased towards a unique equilibrium type, namely fuzzy Nash equilibrium.

Each player i has a membership degree $A_N(i)$ to the Nash rationality (is $A_N(i)$ Nash-biased). The generative relation of fuzzy Nash equilibrium is $\prec_N$.

In the case of a two player game we may describe several fuzzy Nash-Nash equilibria. We may have for instance a $0.2 - 0.99$ fuzzy Nash equilibrium, meaning that $A_N(1) = 0.2$ (player one is 0.2 Nash biased) and $A_N(2) = 0.99$. In this case

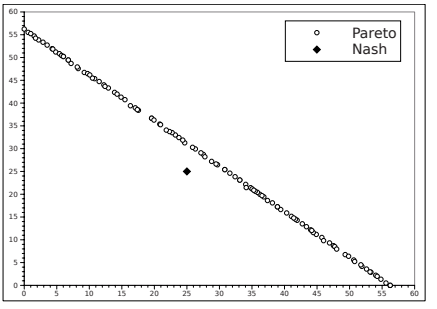


Fig. 1. Detected sharp Nash equilibrium and Pareto front for symmetric Cournot game with two players

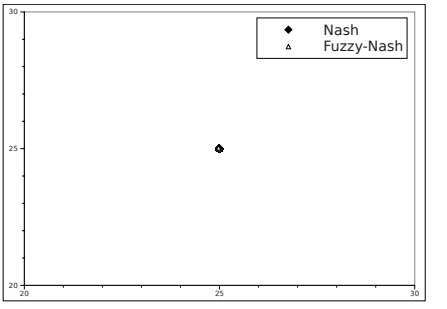


Fig. 2. Detected Fuzzy Nash equilibrium with $A_N(1),\ A_N(2) \in (0, 1]$. It is very close to sharp Nash equilibrium.

the first player is not very interested in playing Nash strategy while the second player is very close to pure Nash rationality.

Let us consider the case of players having an equal bias towards Nash equilibrium:

$$A_N(1) = A_N(2) \in (0, 1].$$

The detected equilibria even at the slightest bias of the players towards Nash is just the classical Nash equilibrium, as depicted in Figure 2.

6.4 Fuzzy Nash-Pareto Equilibrium Detection

Several numerical experiments have been conducted for the Fuzzy Nash-Pareto equilibria, with different memberships degrees.

Consider both players to be Nash biased with 0.8 membership degree and Pareto with 0.2. In this case a small tendency to deviate from pure Nash equilibria towards the Pareto front is detected (Figure 3a and Figure 3b).

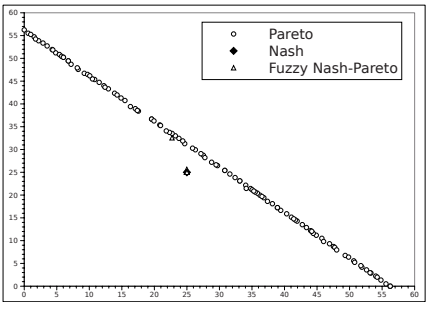


Fig. 3a. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.8$, $A_P(1) = 0.2$, $A_N(2) = 0.8$, $A_P(2) = 0.2$

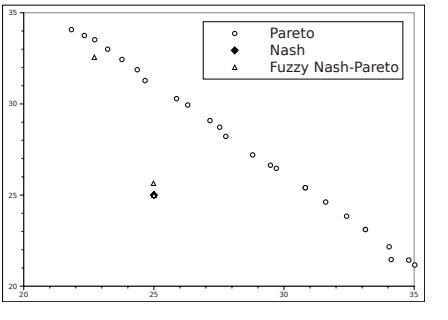


Fig. 3b. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.8$, $A_P(1) = 0.2$, $A_N(2) = 0.8$, $A_P(2) = 0.2$. Detail

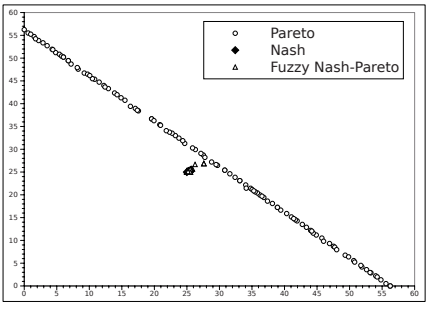


Fig. 4a. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.5$, $A_P(1) = 0.5$, $A_N(2) = 0.5$, $A_P(2) = 0.5$

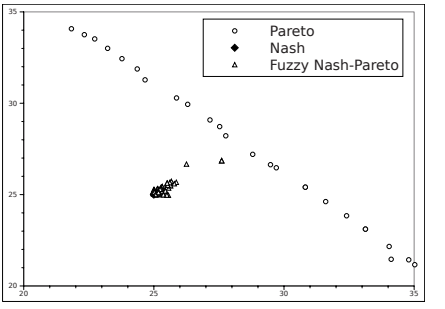


Fig. 4b. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.5$, $A_P(1) = 0.5$, $A_N(2) = 0.5$, $A_P(2) = 0.5$. Detail

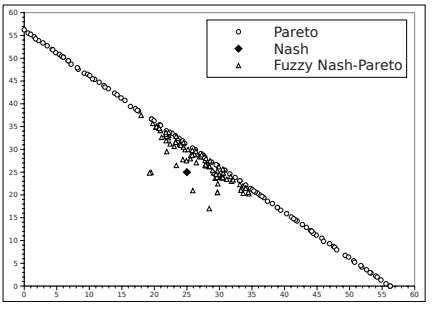


Fig. 5a. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.25$, $A_P(1) = 0.75$, $A_N(2) = 0.25$, $A_P(2) = 0.75$

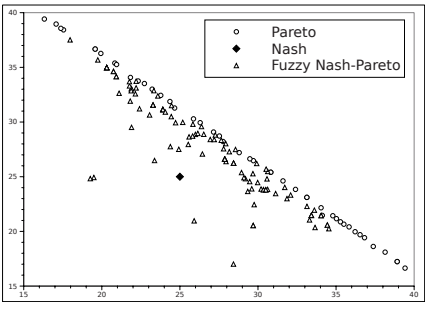


Fig. 5b. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.25$, $A_P(1) = 0.75$, $A_N(2) = 0.25$, $A_P(2) = 0.75$. Detail

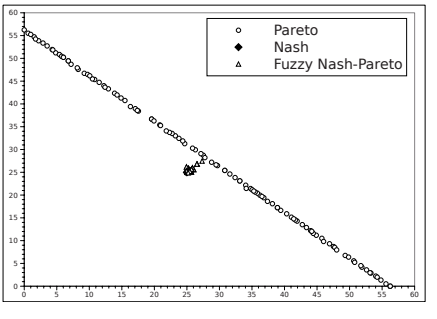


Fig. 6a. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.8$, $A_P(1) = 0.2$, $A_N(2) = 0.2$, $A_P(2) = 0.8$

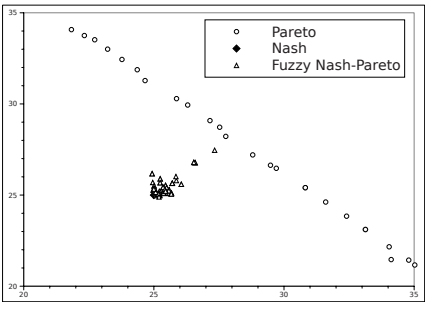


Fig. 6b. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.8$, $A_P(1) = 0.2$, $A_N(2) = 0.2$, $A_P(2) = 0.8$. Detail

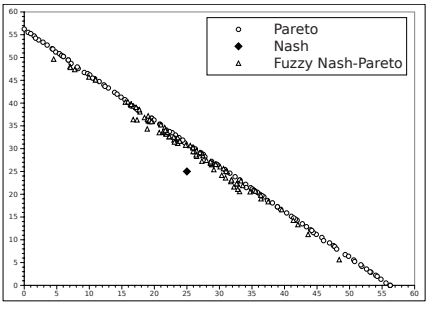


Fig. 7. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.7$, $A_P(1) = 0.3$, $A_N(2) = 0.1$, $A_P(2) = 0.9$

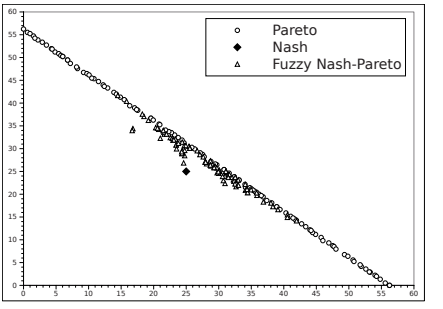


Fig. 8. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.4$, $A_P(1) = 0.6$, $A_N(2) = 0.3$, $A_P(2) = 0.7$

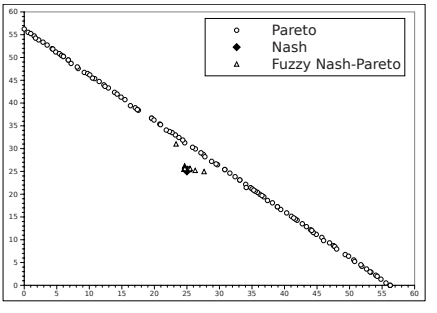


Fig. 9a. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.9$, $A_P(1) = 0.1$, $A_N(2) = 0.4$, $A_P(2) = 0.6$

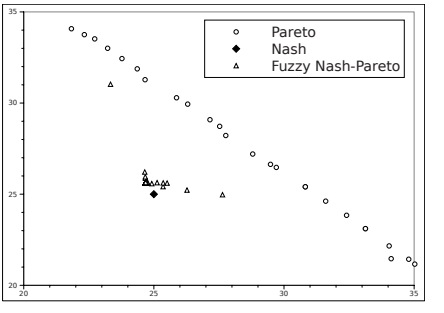


Fig. 9b. Detected Fuzzy Nash-Pareto equilibrium with $A_N(1) = 0.9$, $A_P(1) = 0.1$, $A_N(2) = 0.4$, $A_P(2) = 0.6$. Detail

Increasing players Pareto membership degrees to 0.5 this tendency becomes stronger (Figures 4a and 4b).

The detected Fuzzy Nash-Pareto equlibrium for players with Pareto membership degree 0.75 is depicted in Figure 5a and Figure 5b. The strategies payoffs are spread between the Nash equilibrium and the Pareto front.

An interesting phenomenon arises when the two players have different Nash Pareto membership degrees. In Figures 6a, 6b, 7, 8, 9a, 9b the payoffs for the detected equilibria in such cases are depicted. We denote the tendency of the detected equilibria to spread between the Nash and the Pareto equilibria.

7 Conclusions

The paper represents one step in our program of making game theory more realistic, robust and computationally effective.

We consider the players may be endowed with several types of (fuzzy) rationality. The rationality type expresses the measure a player is biased towards a certain type of equilibrium. This biased may be expressed as a fuzzy set on the set of players. Generative relations for several fuzzy equilibria are defined.

An evolutionary procedure is used to detect fuzzy equilibria in non-cooperative games. Non-domination based fitness assignment is considered. Numerical experiments indicate that the proposed approach may be effective in solving computationally difficult problems related to fuzzy equilibria detection.

Acknowledgements

This research is supported partially by the CNCSIS Grant ID508 *New Computational paradigms for dynamic complex problems* funded by the MEC, and from the Sectoral Operational Programme Human Resources Development, Contract POSDRU 6/1.5/S/3 *Doctoral studies: through science towards society*, Babeş - Bolyai University, Cluj - Napoca, România.

References

1. Bade, S., Haeringer, G., Renou, L.: More strategies, more Nash equilibria, Working Paper 2004-15, School of Economics University of Adelaide University (2004)
2. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II KanGAL Report No. 200001, Indian Institute of Tehnology Kanpur (2000)
3. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester (2001)
4. Dumitrescu, D., Lung, R.I., Mihoc, T.D.: Evolutionary Equilibria Detection in Non-cooperative Games. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoStar 2009. LNCS, vol. 5484, pp. 253–262. Springer, Heidelberg (2009)
5. Lung, R.I., Dumitrescu, D.: Computing Nash Equilibria by Means of Evolutionary Computation. Int. J. of Computers, Communications & Control, 364–368 (2008)
6. Maskin, E.: The theory of implementation in Nash equilibrium: A survey. In: Hurwicz, L., Schmeidler, D., Sonnenschein, H. (eds.) Social Goals and Social Organization, pp. 173–204. Cambridge University Press, Cambridge (1985)
7. McKelvey, R.D., McLennan, A.: Computation of equilibria in finite games. In: Amman, H.M., Kendrick, D.A., Rust, J. (eds.) Handbook of Computational Economics. Elsevier, Amsterdam (1996)
8. Nash., J.F.: Non-cooperative games. Annals of Mathematics 54, 286–295 (1951)
9. Osborne, M.J.: An Introduction to Game Theory. Oxford University Press, New-York (2004)
10. Wu, S.H., Soo, V.W.: A Fuzzy Game Theoretic Approach to Multi-Agent Coordination. In: Ishida, T. (ed.) PRIMA 1998. LNCS (LNAI), vol. 1599, pp. 76–87. Springer, Heidelberg (1999)

An Evolutionary Approach for Solving the Rubik's Cube Incorporating Exact Methods

Nail El-Sourani, Sascha Hauke, and Markus Borschbach

University of Applied Sciences,
Faculty of Computer Science, Chair of Optimized Systems,
Hauptstr. 2, D-51465 Bergisch Gladbach, Germany
nail@elsourani.com, sascha.hauke@fhdw.de, markus.borschbach@fhdw.de

Abstract. Solutions calculated by Evolutionary Algorithms have come to surpass exact methods for solving various problems. The Rubik's Cube multiobjective optimization problem is one such area. In this work we present an evolutionary approach to solve the Rubik's Cube with a low number of moves by building upon the classic Thistlethwaite's approach. We provide a group theoretic analysis of the subproblem complexity induced by Thistlethwaite's group transitions and design an Evolutionary Algorithm from the ground up including detailed derivation of our custom fitness functions. The implementation resulting from these observations is thoroughly tested for integrity and random scrambles, revealing performance that is competitive with exact methods without the need for pre-calculated lookup-tables.

1 Introduction

Solving the Rubik's Cube is a challenging task. Both the size of the solution space induced by the number of attainable states and multiple desirable side-objectives next to restoring the Cube (favorably in the smallest possible number of moves and lowest calculation complexity) make this an interesting optimization problem. Although invented in 1974, the number of moves required to solve any state of Rubik's Cube (the so-called *God's Number*) is yet to be determined after 30 years.

Various algorithms were devised to decrease the upper bound. However, all those approaches are strictly exact methods and the most recent ones rely on terabytes of pre-calculated lookup-tables. This is reflected by the current lowest upper bound of 22 moves achieved by Rokicki [11]. This number was attained by applying the same method he had used earlier for pushing the upper bound to 26, 25 and then 23 moves - using the very same algorithm only on more powerful hardware and a longer calculation time [10], [11].

Evolutionary Algorithms have been successfully applied in a variety of fields, especially highly complex optimization problems [2], [8], [14]. Oftentimes, superior solutions - as compared to classical algorithms have been achieved - notably in multiobjective cases (for example multiconstraint knapsack problems [4]). This gives rise to the idea of applying Evolutionary Algorithms to the Rubik's Cube

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 80–89, 2010.

problem. All relevant approaches are based on dividing the solution space of the Rubik's Cube into mathematical groups, starting with Thistlethwaite using 4 [13], then Reid combining two of Thistlethwaite's groups resulting in total of 3 [9] and finally Kociemba's [7] and Rokicki's approach using 2 subgroups. This makes the group theoretic approach a reasonable starting point for designing Evolutionary Algorithms. It is of particular interest to us to determine how such an EA can solve the Cube without relying on extensive lookup-tables.

2 Notation and Basic Concepts

2.1 Rubik's Cube

The subject of this paper is the classic 3^3 Rubik's Cube. It consists of 26 pieces called cubies: 8 corner cubies, 12 edge cubies and 6 center cubies, distributed equally on the six sides of the Cube. Each side of the Cube is called *face*, each 2-dimensional square on a face is referred to as *facelet*.

A corner cubie shows 3 facelets, an edge 2 and a center 1. Each side of the Cube can be rotated clockwise (CW) and counterclockwise (CCW). Each single move changes the position of 4 edges and 4 corners. The center facelets remain fixed in position. They determined their face's color.

For each edge and corner we distinguish between *position* and *orientation*: i.e. an edge can be in its right position (defined by the two adjacent center colors) but in the wrong orientation (flipped).

There are several known notations for applying single moves on the Rubik's Cube. We will use F, R, U, B, L, D to denote a clockwise quarter-turn of the front, right, up, back, left, down face and Fi, Ri, Ui, Bi, Li, Di for a counterclockwise quarter-turn. Every such turn is a *single move*. In Cube related research, half-turns ($F2, R2, U2, B2, L2, D2$) are also counted as single moves. This notation is independent of colors but depends on the viewpoint. A sequence of moves, an operation, is created by concatenating single moves and is called *operation* (i.e. $FRBiL2$).

2.2 Applied Group Theory

A *group* G is a set together with multiplication and identity e ($eg = g$), inverse ($gg^{-1} = g^{-1}g = e$) and an associative law. A *subgroup* $H < G$ is a subset H that is closed under group operations. $S \subseteq G$, written $G =< S >$ is a *generator* of G if any element of G can be written as a product of elements of S and their inverses. The *order* of the group is the number of elements in it, $|G|$. Given a group G and a subgroup $H < G$, a *coset* of H is the set $Hg = hg : h \in H$; thus, $H < G$ partitions G into cosets. The set of all cosets is written $H \backslash G$.

Obviously, all possible states of a Rubik's Cube are described by the group generated by its applicable moves $G_C =< F, R, U, B, L, D >$, also called the *Cube Group* ($|G_C| = 4.3 \cdot 10^{19}$). Let $H =< L, R, F, B, U2, D2 >$ be a subgroup of G_C, representing a Cube where only the edge positions matter, as no edge orientations can be altered. Thus, $H \backslash G_C$ depicts the left coset space which

contains all possibly attainable states when only flipping edge cubies (changing an edges orientation). For extended explanation refer to [5], [12].

3 Related Work

3.1 Non-evolutionary Approaches

There are several computational approaches for solving the Rubik's Cube, the three most important being the work of Thistlethwaite, Kociemba and Rokicki. Their advanced algorithms are based on group theory concepts and apply advanced concepts such as symmetry cancelation and dedicated traversal methods (e.g. Iterative Deep Searching, IDA*).

Thistlethwaite's Algorithm (TWA) works by dividing the problem into 4 subproblems - specifically subgroups and subsequently solving those. By using precalculated lookup-tables, sequences are put together that move a Cube from one group into another until it is solved [13].

Kociemba's Algorithm takes the idea of dividing the problem into subgroups from Thistlethwaite, but reduces the number of needed subgroups to only 2. This method uses an advanced implementation of IDA*, generating small maps, calculating and removing symmetries from the search tree and tends to solve the Cube close to the shortest number of moves possible. Kociemba made his approach available in form of a program called *Cube Explorer* which can be found at [7].

Rokicki realised that the initial parts of the pathways computed by Kociemba's Algorithm are solutions to a large set of related configurations. He exploits this property by dividing the problem into 2 billion cosets, each containing around 20 billion related configurations. With this method he was able to push the upper bound to 22 moves sufficing to solve the Cube from any initial scrambled configuration [10], [11].

3.2 Evolutionary Approaches

Only a few evolutionary approaches dedicated to solving the Rubik's Cube exist. In 1994 Herdy devised a method which successfully solves the Cube [6] using pre-defined sequences as mutation operators that only alter few cubies, resulting in very long solutions. Another approach by Castella could not be verified due to a lack of documentation. Recently Borschbach and Grelle [1] devised a 3-stage Genetic Algorithm based on a common human "SpeedCubing" method, first transforming the Cube into a 2x2x3 solved state, then into a subgroup where it can be completed using only two adjacent faces (two-generator group).

4 Thistlethwaite's Algorithm

The basic idea of the TWA is to divide the problem of solving the Cube into four independent subproblems by using the following four nested groups: $G_0 =<$

$F, R, U, B, L, D >, G_1 =< F, U, B, D, R2, L2 >, G_2 =< U, D, R2, L2, F2, B2 >$, $G_3 =< F2, R2, U2, B2, L2, D2 >, G_4 = I$. Obviously, $G_0 = G_C$. The functional principle of Thistlethwaite's Algorithm is to put the Cube into a state where it can be solved by only using moves from G_i which again has to be achieved by only using moves from G_{i-1} for $i = 1, \ldots 4$, thus named *nested groups*.

Specifically, every stage of the algorithm is simply a lookup table showing a transition sequence for each element in the current coset space $G_{i+1} \diagdown G_i$ to the next one ($i = i+1$). These coset spaces, each describing a reduced form of the 3^3 Rubik's Cube puzzle, induce different kinds of constraints. This directly results in the total number of attainable states being reduced by using only moves from some subgroup G_{i+1}. The exact orders for each group are calculated as follows:

G_0 $|G_0| = 4.33 \cdot 10^{19}$ represents the order of the Cube Group.

G_1 The first coset space $G_1 \diagdown G_0$ contains all Cube states, where the edge orientation does not matter. This is due to the impossibility of flipping edge cubies when only using moves from G_1. As there are 2^{11} possible edge orientations,

$$|G_1 \diagdown G_0| = 2^{11} = 2048 \tag{1}$$

the order of $|G_1|$ is

$$|G_1| \equiv \frac{|G_0|}{|G_1 \diagdown G_0|} = 2.11 \cdot 10^{16} \, . \tag{2}$$

G_2 Using only moves from G_2, no corner orientations can be altered (eliminating 3^7 states). Additionally, no edge cubies can be transported to or from the middle layer (eliminating $\frac{12!}{(8! \cdot 4!)}$ states). The coset space $G_2 \diagdown G_1$ thus depicts a reduced puzzle of the order

$$|G_2 \diagdown G_1| = 3^7 \cdot \frac{12!}{(8! \cdot 4!)} = 1082565 \tag{3}$$

and

$$|G_2| \equiv \frac{|G_1|}{|G_2 \diagdown G_1|} = 1.95 \cdot 10^{10} \, . \tag{4}$$

G_3 Once in the coset space $G_3 \diagdown G_2$, the Cube can be solved by only using moves from G_3, here the edge cubies in the L, R layers can not transfer to another layer (eliminating $\frac{8!}{(4! \cdot 4!)} \cdot 2$ states) and corners are put into their correct orbits, eliminating $\frac{8!}{(4! \cdot 4!)} \cdot 3$ states). Thus,

$$|G_3 \diagdown G_2| = (\frac{8!}{(4! \cdot 4!)})^2 \cdot 2 \cdot 3 = 29400 \tag{5}$$

and

$$|G_3| \equiv \frac{|G_2|}{|G_3 \diagdown G_2|} = 6.63 \cdot 10^5 \, . \tag{6}$$

G_4 As G_4 represents the solved state - obviously $|G_4| = 1$ which means there exist a mere $|G_3|$ possible states for which a solution needs to be calculated to transfer from $G_4 \setminus G_3$ to solved state.

Most essential to TWA are the groups G_1, G_2, G_3 as G_0 simply describing the Cube Group G_C and G_4 the solved state. To further exemplify how the coset spaces simplify the Rubik's Cube puzzle the following may prove helpful. When looking at the constraints induced by $G_2 \setminus G_1 \setminus G_0$ carefully (combining the constraints induced by $G_2 \setminus G_1$ and $G_1 \setminus G_2$) it is essentially a Rubik's Cube with only 3 colors - counting two opposing colors as one. This representation can be reached for each distinct coset space by examining and applying its effect to the complete Rubik's Cube puzzle.

While solving the Rubik's Cube in a divide and conquer manner, breaking it down into smaller problems (by generating groups and coset spaces) is effective, there exists one major flaw. Final results obtained by concatenating shortest subgroup solution do not necessarily lead to the shortest solution, globally.

5 The Thistlethwaite ES - An Evolution Strategy Based on the Thistlethwaite's Algorithm

As seen above, in the classic TWA the order of each subproblem is significantly smaller than $|G_C|$ and is reduced from stage to stage. The four resulting problem spaces are much more suitable to be solved via ES, as calculation complexity and the probability of ending up in local minima is decreased. Further, this enables the use of truly random mutation operators (otherwise highly ineffective in all of $|G_C|$ [1]) opposed to the hard-coded sequence approach used by Herdy [6].

Thus, we present a 4-phase ES with each phase calculating one group transition (will be referred to as Thistlethwaite Evolution Strategy, TWES). These phases share the same basic selection method but differ in mutation operators and fitness functions. Effectively, the presented ES can best be described as four consecutive ES, each using the solution of the previous one as starting individual to be duplicated (the first using the scrambled input Cube).

5.1 Basic Workflow

A scrambled Cube is duplicated λ times and the main loop is started using a fitness function $phase_0$ and only mutation sequences from G_0. As soon as μ Cubes which solve $phase_0$ have been evolved, the phase transition begins.

During phase transition, from those μ $phase_0$-solving Cubes, a random Cube is chosen and duplicated. This is repeated λ times and yields in the first population after the phase transition. Now the phase-counter is increased by one, and the main ES loop is entered again. This process is repeated until $phase_4$ is solved (i.e. $phase_5$ is reached), presenting a solution sequence to the originally scrambled Cube. This workflow is demonstrated in Fig. 1 (for in-depth implementation details see [3]).

In order to avoid the TWES getting stuck in local optima an upper bound for calculated generations is introduced. As soon as this upper bound is reached,

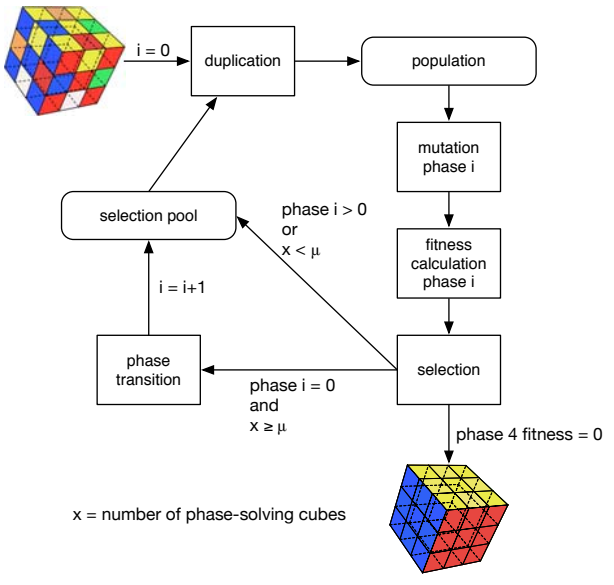


Fig. 1. Basic workflow of Thistlethwaite ES, $i = 0, \ldots, 5$

the TWES resets itself and starts over again. Based on testing several scrambles, the default upper bound is set to 150 generations.

5.2 Rubik's Cube as an Individual

The Rubik's Cube is represented using 6 2D matrices containing values from 1 to 6, each representing one color. Every quarter- and half-turn can be applied to this representation, yielding a total of 18 different single moves while still leaving the Cube's integrity intact.

Thus, mutation is easily realized by not modifying a single facelet's color but applying a sequence of moves to the Cube. This guarantees that the Cube's integrity stays intact at all times and makes a separate integrity test superfluous.

Every individual remembers the mutations it has undergone, i.e. a list of moves that have been applied. To keep this list as small as possible, redundant moves are automatically removed. For example an individual that has been mutated with F and is then mutated with $FRRiB$ will only remember the optimized sequence $F \cdot FRRiB = F2B$, preventing redundancy. Essentially, this is realized via a while-loop, eliminating redundant moves in each pass until no further optimizations can be made: e.g. $F2BBiR2R2F$ is optimized to Fi by first removing BBi, then removing $R2R2$ and finally transforming $F2F$ into Fi.

5.3 Fitness Function

Translating the TWA into an appropriate Fitness Function for an Evolutionary Algorithm essentially forces the design of four distinct subfunctions. As each

subgroup of G_0 has different constraints, custom methods to satisfy these constraints are proposed.

$G_0 \rightarrow G_1$ To reach G_1 from any scrambled Cube, we have to orient all edge pieces right while ignoring their position. The fitness function for this phase simply increases the variable $phase_0$ by 2 for each wrong oriented edge. Furthermore, we add the number of moves that have already been applied to the particular individual in order to promote shorter solutions. Finally, we adjust the weight between w (number of wrong *oriented* edges) and c (number of moves applied to current Cube individual). This will be done similarly in all subsequent phases.

$$phase_0 = 5 \cdot (2w) + c \tag{7}$$

With a total of 12 edges which can all have the wrong orientation this gives $max\{2w\} = 24$. The Cube has been successfully put into G_1 when $phase_0 = c$. Reaching G_1 is fairly easy to accomplish, thus making the weight-factor 5 a good choice.

$G_1 \rightarrow G_2$ In order to fulfill G_2 the 8 corners have to be oriented correctly. Edges that belong in the middle layer get transferred there. Tests with the Thistlethwaite ES showed it somewhat problematic to do this in one step. Oftentimes, the algorithm would get stuck in local optima. To solve this, the process of transferring a Cube from G_1 to G_2 has been divided into two parts. First, edges that belong into the middle layer are transferred there. Second, the corners are oriented the right way. The first part is fairly easy and the fitness function is similar to that from $phase_0$ except for w (number of wrong *positioned* edges), i.e. edges that should be in the middle layer but are not.

$$phase_1 = 5 \cdot (2w) + c \tag{8}$$

In the second part, for each wrong positioned corner, 4 penalty points are assigned as they are more complex to correct than edges. Obviously, in order to put the Cube from G_1 to G_2 both phases described here have to be fulfilled, which yields:

$$phase_2 = 10 \cdot (4v) + phase_1 \tag{9}$$

where v represents the number of wrong *oriented* corners. The weighing factor is increased from 5 to 10 to promote a successful transformation into G_2 over a short sequence of moves.

$G_2 \rightarrow G_3$ We now have to put the remaining 8 edges in their correct orbit. The same is done for the 8 corners which also need to be aligned the right way. Thus, the colors of two adjacent corners in one circuit have to match on two faces. In G_3 the Cube will only have opposite colors on each face. Let x (number of wrong *colored* facelets) and y (number of wrong *aligned* corners), then

$$phase_3 = 5 \cdot (x + 2 \cdot y) + c\,. \tag{10}$$

$\mathbf{G_3} \rightarrow \mathbf{G_4}(\mathbf{solved})$ The Cube can now be solved by only using half-turns. For the fitness function we simply count wrong colored facelets. Let z be the number of wrong colored facelets, then

$$phase_4 = 5 \cdot z + c \,. \tag{11}$$

To summarize, 5 different fitness functions are needed for the Thistlethwaite ES. $phase_i$ is solved if $phase_i = c$, $i = 0, ..., 4$ and with the properties of nested groups we can conclude, given the above, a solved Cube implies:

$$\sum_{0}^{4} phase_i = c \,. \tag{12}$$

Fulfilling the above equation satisfies the constraints induced by the groups $G_0, \ldots, G_4$, with the final fitness value c describing the final solution sequence length. The weight factors chosen are based on consecutive testing throughout development. The ratio is dictated by the size of the nested groups. Finding optimal weights presents a seperate optimization problem and may be subject to future work.

5.4 Mutation Operators

The mutation operators are dictated by the subgroups used. Conveniently, the maximum sequence length (s) needed to transform the Cube from one subgroup to another is given by Thistlethwaite [13]. Those lengths are 7,13,15,17 (the sum of which is 52, hence "52 Move Strategy") for each group transition respectively. An individual in phase i is mutated by:

1. generating a random length (l) with $0 \leq l \leq s$, according to i ($i = 0 \rightarrow s = 7, i = 1 \rightarrow s = 13, i = 2, 3 \rightarrow s = 15, i = 4 \rightarrow s = 17$)
2. concatinating l random single moves from the according group G_i
3. applying this sequence to the current Cube individual

For example: Let $i = 2$ (transitioning from $G_2 \rightarrow G_3$). The maximum sequence length for this step is $s = 15$. Let random $l = 4, (0 \leq 4 \leq 15)$. Next, we chose a random single move from G_2, repeat this a total of 4 times and concatinate these to form a sequence. Let those 4 single moves be $D, F2, R2, U$. This results in the sequence $DF2R2U$ representing the present mutation which is applied to the current Cube individual.

In case of $l = 0$ the mutation is an empty sequence, leaving the current individual untouched. Given an appropriate fitness, this allows distinct Cubes to survive multiple generations.

5.5 Selection Method

The selection method used is a modified truncation selection. The selection pool is generated by chosing the μ best individuals from the current population. Therefrom, each individual is duplicated with the same probability $\frac{1}{\mu}$, repeated λ times

to form the new population. This approach implicates a low selection pressure and hence favors a higher diversity, as it is a key importance to enable the survival of alleged suboptimal individuals. The combination of random mutation operators, phase transitions and redundant move removal can result in an abrupt fitness improvement of such individuals.

Furthermore, similar to Kociemba's key idea of continuing calculation after some solution to one phase has been found [7], our ES continues until μ *different* such individuals have been evolved. These are then duplicated as described above to form the inital population for the subsequent phase. Put simply, $phase_{i+1}$ starts with a population pool of λ $phase_i$-solving Cubes of high diversity. This can have a positive effect on overall solution length and calculation time, as remarked by Kociemba [7] and counters the major flaw of the classic TWA mentioned in section 4.

6 Benchmarks

To provide a brief performance overview 100 random scrambles of minimum length 10 and maximum length 50 were generated and and solved in 5 repetitions. Solution lengths and calculation time are of particular interest to us. The test was conducted with the TWES using $(\mu, \lambda) = (1000, 50000)$, weighing factors $(5, 5, 5, 5, 5)$, mutation lengths $(5, 5, 13, 15, 17)$ and maximum generations before reset (250).

Table 1. Solutions of 100 random scrambles, 5 repetitions, Thistlethwaite ES

	Run 1	Run 2	Run 3	Run 4	Run 5
avg. Generations	95.72	100.63	92.71	99.66	92.22
avg. Moves	50.67	50.32	50.87	50.23	49.46
avg. Time(s)	321.78	381.68	393.99	312.98	287.93

As seen in Table 1, the solution sequences hit an average of about 50 single moves, further demonstrating a consistent performance throughout the repetitions. Most scrambles are solved in 35-45 moves, outliers are responsible for the higher average count. Extensive additional benchmarks can be found in [3].

7 Conclusion

The benchmarks are promising, yielding comparable results to the classic TWA. Outliers calculated by TWES provide both significantly shorter and longer solutions. This is most probably due to inter-group dependencies and future focus lies on increasing our TWES' tendency to such shorter results. Instead of obtaining static solutions dictated by the lookup-table used in the classic TWA, the dynamic evolution process enables those shorter solution sequences not previously possible.

Regarding the Rubik's Cube optimization problem, our evolutionary approach is evidently competitive with the exact method it adepts. As this was the first such attempt - based on the first group theoretic exact approach using lookup-tables (Thistlethwaite) - future work promises further improvement. This algorithm only solves the classic 3^3 Rubik's Cube, just as the exact method it is based on does. However, our modular EA can also be used to solve higher dimensional Rubik's Cubes by appropriately substituting the current fitness functions.

The next developmental step will adept approaches that reduce the number of subgroups to 3 and then 2, potentially yielding further improvement in solution sequence length. Conveniently, our implementation already provides such possibilities for extensions, enabling quick testing of different subgroup combinations.

References

1. Borschbach, M., Grelle, C.: Empirical Benchmarks of a Genetic Algorithm Incorporating Human Strategies. Technical Report, University of Applied Sciences, Bergisch Gladbach (2009)
2. Boyzejko, W., Wodecki, M.: A Hybrid Evolutionary Algorithm for some Discrete Optimization Problems. In: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, pp. 326–331. IEEE Computer Society, Washington (2005)
3. El-Sourani, N.: Design and Benchmark of different Evolutionary Approaches to Solve the Rubiks Cube as a Discrete Optimization Problem. Diploma Thesis, WWU Muenster, Germany (2009)
4. Florios, K., Mavrotas, G., Diakoulaki, D.: Solving Multiobjective, Multiconstraint Knapsack Problems Using Mathematical Programming and Evolutionary Algorithms. European Journal of Operational Research 203, 14–21 (2009)
5. Frey, A., Singmaster, D.: Handbook of Cubic Math. Enslow, Hillside (1982)
6. Herdy, M., Patone, G.: Evolution Strategy in Action, 10 ES-Demonstrations. Technical Report, International Conference on Evolutionary Computation (1994)
7. Kociemba, H.: Cube Explorer, http://kociemba.org/Cube.htm
8. Muehlenbein, H., Mahnig, T.: FDA - A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions. Evol. Comput. 7, 353–376 (1999)
9. Reid, M.: Cube Lovers Mailing List, http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/Index_by_Author.html
10. Rokicki, T.: Twenty-Five Moves Suffice for Rubik's Cube, http://Cubezzz.homelinux.org/drupal/?q=node/view/121
11. Rokicki, T.: http://www.newscientist.com/article/mg19926681.800-cracking-the-last-mystery-of-the-rubiks-Cube.html
12. Singmaster, D.: Notes on Rubik's Magic Cube. Enslow, Hillside (1981)
13. Thistlethwaite, M.B.: The 45-52 Move Strategy. London CL VIII (1981)
14. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Penn State (1999)

Evolution of Artificial Terrains for Video Games Based on Accessibility

Miguel Frade[1], Francisco Fernandez de Vega[2], and Carlos Cotta[3]

[1] Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Portugal
mfrade@estg.ipleiria.pt
[2] Centro Universitario de Mérida, Universidad de Extremadura, Spain
fcofdez@unex.es
[3] ETSI Informática, Campus de Teatinos, Universidad de Málaga, Spain
ccottap@lcc.uma.es

Abstract. Diverse methods have been developed to generate terrains under constraints to control terrain features, but most of them use strict restrictions. However, there are situations were more flexible restrictions are sufficient, such as ensuring that terrains have enough accessible area, which is an important trait for video games. The Genetic Terrain Program technique, based on genetic programming, was used to automatically evolve Terrain Programs (TPs - which are able to generate terrains procedurally) for the desired accessibility parameters. Results showed that the accessibility parameters have negligible influence on the evolutionary system and that the terminal set has a major role on the terrain look. TPs produced this way are already being used on *Chapas* video game.

Keywords: genetic terrain programming, artificial terrains, video games.

1 Introduction

Generating artificial terrains is an important topic in computer graphics, specially in video games where its application is probably more prominent. Some of the most popular techniques are fractal algorithms. These algorithms are the favorite ones by game designers, mainly due to their speed and simplicity of implementation. However, procedural techniques, like fractals, have also their own limitations [5]. The main disadvantage is the difficulty of modelling with them. It is very difficult or impossible to know how to modify them to achieve a certain local effect.

Diverse methods have been developed to produce terrains under constraints in attempt to control terrain features. The technique in [12] consists in generating a terrain model by computing the interpolation of point clouds or contour lines. The authors of [10] managed to obtain good approximations of downsampled elevation datasets using radial basis functions. Other methods use patches [13], small-scale [3] or microscopic [4] features of existing terrains to synthesize new ones that satisfy the user global constraints. The method presented in [11] deforms an initial fractal terrain by applying local modifications to satisfy a set

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 90–99, 2010.

of various fixed constraints. A constrained fractal model based on midpoint displacement algorithm is presented in [1]. However, most of these methods are focused on the reconstruction of Data Elevation Models and use only height constraints, besides many of them suffer from either time and/or manipulation complexity.

To use height constraints the designer must specify both the height value and its location. However, there might be situations were less strict restrictions are preferable. For instance, a designer might want a nearly flat, or mountainous terrain, without caring about the specific location or size of terrain features. A two step approach, designated Genetic Terrain Programming (GTP), was proposed in [6], [7] that allows the generation of terrains with the desired features or aesthetic appeal without formal restrictions. The first step consists of interactive evolution, with genetic programming, of mathematical expressions designated TPs (Terrain Programs). On the second step TPs are computed to generate height maps. Due to the interactive nature of this approach, from now on it will be abbreviated as GTP_i. Nevertheless, there are terrain features that are better evaluated by computers due to their numerical nature. Besides, the search for a terrain with very specific features might be a tiresome endeavor on interactive evolutionary applications [2].

Accessibility is a crucial structural feature that highly influences the attainable movements of characters/avatars around the video game terrain, as well as the potential placement of buildings and other structures. The author of [9] employs accessibility restrictions to generate terrains, but the used technique may not fully meet the constraints. On this paper it is presented a new version of GTP that performs automatic evaluation of terrains based on accessibility constraints, which will be designated from now on as GTP_a.

Section 2 introduces some background about the generation of height maps from TPs. The fitness function and the reasoning behind it are also detailed on section 3. A set of tests were conducted and the used parameters, as well as its results are showed in Section 4. Finally, the conclusions and future work are presented on Section 5.

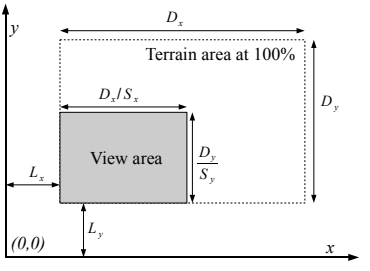
2 Background

An important characteristic of procedural techniques is their ability to generate a scene with the required resolution and zoom level. This is probably the main advantage of the procedural techniques over other techniques. In spite of the procedural nature of TPs, the implementation in GTP_i only allowed control over terrain resolution, not zoom level. This is a limitation that runs against the procedural advantages. Therefore, in GTP_a both the terminal and function sets were modified in order to generate TPs that permit also the control over zoom. The function set used in GTP_i was simplified to remove all functions that prevented the zoom feature, Table 1 shows the GTP_a function set.

Three different terminal sets are used on this work to evaluate its influence on the resulting terrains: $T_1 = \{noise(x, y), rec\}$, $T_2 = \{x, y, rec\}$ and

Table 1. GP function set

Name	Description
$plus(a,b)$, $minus(a,b)$, $multiply(a,b)$	arithmetical functions
$sin(a)$, $cos(a)$, $tan(a)$, $atan(a)$	trigonometric functions
$exp(a)$	returns e^a
$myLog(a)$	returns 0 if $a = 0$ and $log(\|a\|)$ otherwise
$myPower(a,b)$	returns 1 if $b = 0$, 0 if $a = 0$ and $\|a\|^b$ otherwise
$myDivide(a,b)$	returns a if $b = 0$ and $a \div b$ otherwise
$mySqrt(a)$	returns $\sqrt{\|a\|}$
$negative(a)$	returns $-a$



$$h_{r,c} = f\left(\frac{c \times \frac{D_x}{n_c - 1}}{S_x} + L_x, \frac{r \times \frac{D_y}{n_r - 1}}{S_y} + L_y\right) . \quad (1)$$
$$r \in \{1, \cdots, n_r\}, c \in \{1, \cdots, n_c\},$$
$$D_x, D_y, S_x, S_y \in \mathbb{R}^+ \text{ and } L_x, L_y \in \mathbb{R}$$

Fig. 1. Terrain view area

$T_3 = \{noise(x,y), x, y, rec\}$, where *rec* stands for *random ephemeral constant*. The $noise(x,y)$ terminal is a stochastic function that is commonly used in fractals. Its ideal properties are [5]: have a repeatable pseudorandom output based on its inputs variables x, y; the output range is known, namely from -1 to 1; when analyzed in the frequency domain, the output is band-limited, with a maximum frequency of about 1; the noise function should not exhibit obvious periodicities or regular patterns; the noise function is stationary, that is, its statistical character should be translationally invariant; and the noise function is isotropic, that is, its statistical character should be rotationally invariant.

TPs are able to generate an unlimited continuous surface. In order to create an height map from a TP, the continuous surface must be sampled with fixed increments of x and y to obtain the corresponding altitude z, where $z = f(x,y)$, $x, y, z \in \mathbb{R}$. The altitudes values are stored in matrix H, whose size $n_r \times n_c$ depends on the amount of samples and therefore define the height map resolution. Equation (1) shows the relation between the height map matrix H and the TPs continuous function obtained with GTP_a. $h_{r,c}$ represents the elevation value for row r and column c, and D_x, D_y are the terrain dimensions. S_x, S_y allow the control of the zoom level and L_x, L_y allow us to localize the origin of the terrain view area (see Fig. 1).

3 Accessibility Score Fitness Function

Movement and structure placing is often restricted to low slopes, therefore, it is necessary to analyze the changes in declination of the terrain. For that purpose the slope map S is defined to store the declination for each cell r, c of the height map H. The slope values are calculated as the magnitude of the gradient vector (tangent vector of the surface pointing in the direction of steepest slope) [8]. With this approach, the slope is computed at a grid point, which depends on the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ of the height map function $z = f(x, y)$. The most common approximation for partial derivatives is a weighted average of the elevation differences between the given point and all points within its 3×3 neighborhood [8]. The estimate of the partial derivatives for cell z_5 (see Fig. 2) are given by (2) and (3), were Δx and Δy are the height map distances between each cell.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$\frac{\partial f}{\partial x} \approx \frac{(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)}{8\Delta x} \ . \tag{2}$$

$$\frac{\partial f}{\partial y} \approx \frac{(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)}{8\Delta y} \ . \tag{3}$$

Fig. 2. Neighbor positions

Player units must be able to move around the terrain and a sufficient number of nearly flat areas must exist for building placement. Therefore, cells with a declination below a certain threshold allowing unit movement should be connected in a large area. To analyze the slope map according to this criteria an accessibility map A is created with the same size of the height map. $A = \{a_{r,c}\}_{c \leq n_c}^{r \leq n_r}$ is a binary map whose cell values depend on the threshold S_t, with $a_{r,c} = 0$ if $s_{r,c} < S_t$ or $a_{r,c} = 1$ if $s_{r,c} \geq S_t$.

The nearly flat areas, whose $s_{r,c} < S_t$, that are not connected to the main area will be inaccessible, so the next step is to search for the biggest connected area in A with a component labeling algorithm. This algorithm returns the amount of connected areas and the number of cells contained within each one. Then, the smaller connected areas are removed from the accessibility map. Next we classify the terrains accordingly to the accessibility value υ given by (4), where $n_r n_c$ is total amount of cells in the height map and C_a is the number of cells contained on the biggest connected area of A. The formula in (4) was built to be minimized, so the smaller value of υ the larger the accessibility area is.

$$\upsilon = \frac{n_r n_c}{C_a}, \quad C_a \neq 0 \ . \tag{4}$$

The accessibility criteria alone would make a completely flat terrain the best fit. However, a completely flat terrain does not add realism or interest to the terrain and does not provide obstacles to units movement, which is undesirable.

To prevent this, we defined the accessibility score υ_s, in (5). The biggest connected area is limited by the threshold υ_t, where $p \in [0,1]$ is the desired area for the biggest connected area. The *ceil* function is used to ensure that the amount of desired cells for the biggest connected area is round up to the nearest integer value. This way it will be possible for υ_s to achieve the exact value of zero and stop the evolutionary process. Otherwise we would have to stipulate a tolerance value within which υ_s would be considered close enough to zero to stop the evolutionary process. However, the tolerance value would be dependent from the chosen resolution for the height map, which is undesirable.

$$\upsilon_s = |\upsilon - \upsilon_t| \,, \quad \text{where} \quad \upsilon_t = \frac{n_r n_c}{\lceil p n_r n_c \rceil}, \quad p \neq 0 \ . \tag{5}$$

4 Tests and Results

A battery of tests was conducted to access the features and realism of terrains produced by terminal sets T_1, T_2 and T_3 evaluated by the accessibility score. We want also to assess the impact of different percentages of accessibility area, thus for each terminal set three different percentages were used: $p = 70\%$, $p = 80\%$ and $p = 90\%$. There are a total of 9 different tests that were performed on a Pentium Core 2 Duo at 2.0 GHz with 1 GB of RAM running GNU/Linux. All tests were conducted with the accessibility threshold set to $S_t = 10\%$ and the parameters shown on Table 2. On the left side of Table 2 are the GP system parameters and on the right side are the parameters used to compute height maps from TPs.

Our main goal are TPs and the terrains they generate, Fig. 3 shows three examples of height maps as gray-scale images (one for each terminal set) and their respective accessibility maps. The second row shows the same terrains, but computed in 3 dimensions with Blender 3D (`www.blender.org`) and rendered with YafaRay (`www.yafaray.org`). These terrains were chosen because they were the ones that presented the shortest TPs, which are displayed in Fig. 4.

The next step was to visually compare and analyze all the terrains produced by the resulting TPs of each terminal set. Our analysis focused on three

Table 2. GP and height map parameters

GP	Value	Height map	Value
maximum generations	50	n_r and n_c	128
population size	500	L_x and L_y	0
initialization method	half and half	S_x and S_y	1
	ramped from 2 to 6	D_x and D_y	10
max. depth	17		
selection operator	tournament, size 7		
crossover operator rate	70%		
mutation operator rate	30%		

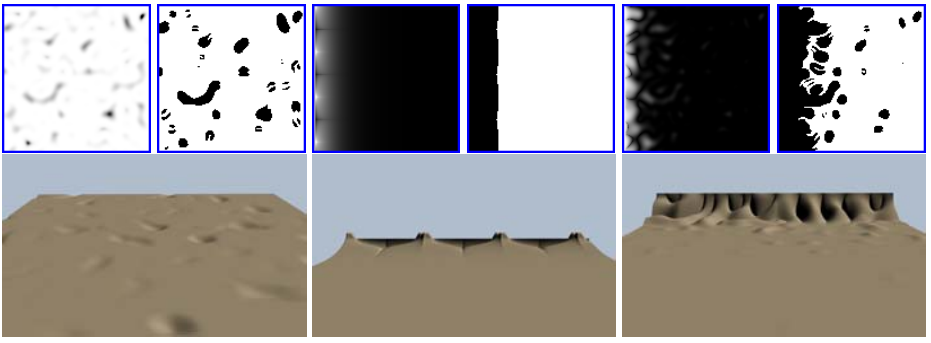


Fig. 3. Examples of height maps (as gray-scale images) and their respective accessibility maps on the first row. The first pair of images were obtained with T_1 (see TP_1 on Fig. 4), the second pair with T_2 (see TP_2) and the third pair with T_3 (see TP_3).The second row has the terrains rendered in 3D.

TP_1 = *sin(sin(cos(myPower(multiply(myNoise(x,y),myNoise(x,y)),minus(exp(atan(multiply(myNoise(x,y), myNoise(x,y))))),sin(myNoise(x,y))))))*

TP_2 = *cos(atan(minus(exp(x),myDivide(multiply(0.07358,0.93756),myDivide(x,myLog(myLog(sin(y))))))))*

TP_3 = *multiply(myPower(myNoise(x,y),x),cos(atan(x)))*

Fig. 4. Terrain programs that generated the terrains on Fig. 3

terrain properties: suitability (to video games), realism and features diversity. The terrains obtained from terminal set T_1 were the ones that showed more usefulness with several obstacles distributed around the terrain area. They also present smooth transitions between the different terrain features and without visible pattern repetition, which give them a natural look. However, apart from a few exceptions, terrains tend to be very similar, like if the locations of the hills and valleys were simply shifted or resized. On the opposite side are terrains produced by terminal set T_2. Terrains tend to exhibit geometric patterns and sudden height changes, which gives them an unnatural look. The origin of the terrain view area seems to have a big impact, because $L_x = L_y = 0$, a value were the discontinuities of our protected functions happens. This suggests the further study of T_2 with a different view area origin values. Terrains obtained with T_2 showed more diversity across the 20 runs than T_1 terrains, but T_2 terrains tend to present a single large obstacle, which makes them less suitable. Our richest terminal set, T_3, was the one that brought out more diverse terrains. On these terrains geometric patterns are occasionally visible, but with much less visual impact than with T_2. Regarding realism T_3, some terrains look more natural than others and a few have also an odd look. To sum up, we classify our terminal sets (from better to worst) regarding realism: T_3, T_1 and T_2; regarding diversity: T_3, T_2 and T_1; finally, regarding suitability: T_1, T_3 and T_2.

Table 3. Average values (and standard deviation of the mean) for 20 runs of GTP_a

Terminal set	T_1			T_2			T_3		
p	70%	80%	90%	70%	80%	90%	70%	80%	90%
Generations	8.25	8.80	8.00	7.95	8.00	9.10	8.40	7.30	7.65
	(0.49)	(0.39)	(0.43)	(0.53)	(0.45)	(0.64)	(0.48)	(0.61)	(0.44)
TP size	44.75	44.35	44.20	48.20	50.25	61.70	39.10	30.80	40.90
	(2.33)	(4.51)	(4.57)	(3.97)	(4.22)	(5.55)	(4.27)	(3.48)	(4.56)
TP depth	13.40	13.15	12.70	12.65	12.30	14.35	11.85	9.30	11.60
	(0.60)	(0.63)	(0.84)	(0.70)	(0.59)	(0.56)	(0.93)	(0.80)	(0.95)
Run time (s)	99.52	112.26	102.82	93.17	88.78	102.87	97.57	77.86	77.78
	(8.50)	(6.05)	(8.68)	(12.56)	(7.84)	(13.47)	(8.93)	(9.12)	(6.46)

The GP system returned also the number of generations, TP size (amount of nodes), TP depth (tree depth) and run time of each run. This data is summarized in Table 3 where the values shown are the average for 20 runs (with different seeds) and the correspondent standard deviation of the mean inside round brackets. The fitness value is not present on this table, because all runs were able to reach the goal of $\upsilon_s = 0$.

Regarding the TPs sizes and depths what stands out is that for terminal set T_1 these values are almost constant for all 3 values of p, for terminal set T_2 they tend to increase with p and that terminal set T_3 is the one that has smaller values. These results show that the richer terminal set T_3 is able to produce shorter and therefore simpler TPs than the other terminal sets. Run times are also smaller for terminal set T_3, except for $p = 70\%$ which is similar to the other terminals. Run times for terminal set T_1 are slightly longer, which was expected because $noise(x, y)$ is far more complex than the other terminals.

The average number of generations across all tests are very similar with values ranging from 7.30 to 9.10. This means that the system is able to reach a fitness value of zero easily and that there are many solutions to reach that goal. The diversity of TPs to have a perfect fitness value is very interesting, because we want to be able to generate a width range of terrains that fit our goal. This raised the question: how different are the resulting terrains from each other when only the seed is changed? To help answer this question we compared each accessibility map with the other 19 from the 20 runs of each test. The comparison consists in measuring how much inaccessible (black) area overlaps, as shown in Fig. 5. An overlap value of 100% would mean that the maps were equal.

For each test the amount of comparisons is the result of combinations without repetitions of $\binom{20}{2} = 190$. The higher quantaties of comparisons returning lower overlap values the bigger are the differences between accessibility maps by changing only the seed. The boxplot on Fig. 6 show that the higher the value of p is, the less the accessibility maps tend to overlap, independently of the used terminal set. This is an expected result, because the inaccessible areas are smaller the larger p is, so there are less probability of overlap. The overlap values from

Fig. 5. Comparison of two accessibility maps obtained with T_1 (first and center images) and the resulting overlap (right image) - the amount of overlap is 50.07%

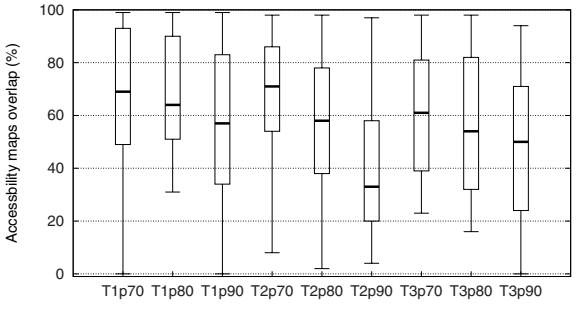


Fig. 6. Boxplot of accessibility maps overlap for each test

Fig. 7. Screenshot of the *Chapas* video game with a terrain generated by a TP

maps obtained with terminal set T_1 tend to be higher than the other ones, which means that terrains will be more alike with each other, which corroborates our visual inspection. For terminal set T_2 the amount of overlap is highly influenced by the value of p, presenting the higher values for $p = 70\%$ and the lowest for $p = 90\%$. Finally, for terminal set T_3 there are lower chances to get similar terrains, which is an advantage in our case. Nevertheless, none of the comparisons, among all tests, returned overlap values of 0% or 100%.

TPs produced by GTP_a are already being used on the video games Chapas, which is still in development (see Fig. 7 and `http://tr.im/chapas`). To generate a terrain with 1024×1024 cells, the fastest TP (from T_3) took 0.1 seconds and the longest (from T_1) took 1.1 seconds. Although these times are in the same

magnitude of the ones obtained by [1], they can be further improved because TPs are easily parallelizable without requiring any interprocess communication.

5 Conclusions

A new version of GTP, GTP$_a$, as been presented that performs automated evaluation of TPs based on accessibility parameters. These parameters allow us to control the slope threshold, that differentiate the accessible from the inaccessible terrain areas, and how much area should be made accessible. Through a series of experiments we have shown that our fitness function, the accessibility score, is able to produce many solutions that fit our goal. Furthermore, the required number of generations is not influenced by changing the terminal set or the amount of desired accessible area. The terminal set can have a great influence on the terrain look, with T_3 producing more realistic terrains, but also some odd looking terrains. T_3 showed us the importance of rich terminal sets to achieve shorter and simpler TPs. T_1 produced the better suitable terrains for a video game. However, our fitness function does not account for terrain realism, requiring a visual inspection before using them. With regard to the previous version of GTP, the new approach allowed to remove the human factor from the evolutionary process and this way avoid designer fatigue. The usefulness of this technique is shown by its integration on the Chapas video game.

Terrain view area is defined by several parameters and some of them, like L_x,L_y, have influence on the result. Further tests must be conducted to access the level and importance of their impact. Slope is just one metric from many that geomorphology uses to classify height maps of real terrains. Other metrics, such as convexity, can be incorporated on the fitness function to allow finner control over terrain features. Multi-objective optimization techniques can be used in this context. Finally, in spite of our interesting results, we have not found yet the perfect terminal set to enable GTP$_a$ to achieve a widther range of real looking terrains types. Further research on this topic is needed.

Acknowledgments. The second author is supported by National Nohnes project TIN2007-68083-C02-01 of Spanish MS. The third author is supported by project TIN2008-05941 of Spanish MICINN. Thanks are due to the reviewers for their detailed and constructive comments.

References

1. Belhadj, F.: Terrain modeling: a constrained fractal model. In: 5th International conference on CG, virtual reality, visualisation and interaction in Africa, pp. 197–204. ACM, Grahamstown (2007)
2. Bentley, P.: Evolutionary Design by Computers. Morgan Kaufmann Publishers, Inc., USA (1999)
3. Brosz, J., Samavati, F.F., Sousa, M.C.: Terrain synthesis by-example. In: First International Conference on Computer Graphics Theory and Applications (2006)

4. Chiang, M., Huang, J., Tai, W., Liu, C., Chang, C.: Terrain synthesis: An interactive approach. In: International Workshop on Advanced Image Tech. (2005)
5. Ebert, D., Musgrave, K., Peachey, D., Perlin, K., Worley, S.: Texturing and Modeling: A Procedural Approach, 3rd edn. Morgan Kaufmann, San Francisco (2003)
6. 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)
7. Frade, M., de Vega, F.F., Cotta, C.: Breeding terrains with genetic terrain programming - the evolution of terrain generators. International Journal for Computer Games Technology 2009, (Article ID 125714) 13 (2009)
8. Horn, B.K.P.: Hill shading and the reflectance map. Proceedings of the IEEE 69(1), 14–47 (1981)
9. Olsen, J.: Realtime procedural terrain generation - realtime synthesis of eroded fractal terrain for use in computer games. Department of Mathematics and Computer Science (IMADA). University of Southern Denmark (2004)
10. Pouderoux, J., Gonzato, J.C., Tobor, I., Guitton, P.: Adaptive hierarchical RBF interpolation for creating smooth digital elevation models. In: GIS 2004 - 12th annual ACM international workshop on Geographic information systems, pp. 232–240. ACM, New York (2004)
11. Stachniak, S., Stuerzlinger, W.: An algorithm for automated fractal terrain deformation. Computer Graphics and Artificial Intelligence 1, 64–76 (2005)
12. Vemuri, B., Mandal, C., Lai, S.H.: A fast gibbs sampler for synthesizing constrained fractals. IEEE Transactions on Visualization and Computer Graphics 3(4), 337–351 (1997)
13. Zhou, H., Sun, J.: Terrain synthesis from digital elevation models. IEEE Transactions on Visualization and Computer Graphics 13(4), 834–848 (2007)

Evolving Behaviour Trees for the Commercial Game DEFCON

Chong-U Lim, Robin Baumgarten, and Simon Colton

Computational Creativity Group
Department of Computing, Imperial College, London
www.doc.ic.ac.uk/ccg

Abstract. Behaviour trees provide the possibility of improving on existing Artificial Intelligence techniques in games by being simple to implement, scalable, able to handle the complexity of games, and modular to improve reusability. This ultimately improves the development process for designing automated game players. We cover here the use of behaviour trees to design and develop an AI-controlled player for the commercial real-time strategy game DEFCON. In particular, we evolved behaviour trees to develop a competitive player which was able to outperform the game's original AI-bot more than 50% of the time. We aim to highlight the potential for evolving behaviour trees as a practical approach to developing AI-bots in games.

1 Introduction

The ability of Artificial Intelligence methods in games to deliver an engaging experience has become an important aspect of game development in the industry, and as such, numerous techniques have been developed in order to deliver realistic game AI. However, as Jeff Orkin remarked, that if the audience of the Game Developers Conference were to be polled on the most common A.I techniques applied to games, one of the top answers would be Finite State Machines (FSMs) [11]. Behaviour trees have been proposed as an improvement over FSMs for designing game AI. Their advantages over traditional AI approaches are being simple to design and implement, scalability when games get larger and more complex, and modularity to aid reusability and portability. Behaviour trees have recently been adopted for controlling AI behaviours in commercial games such as first-person-shooter Halo2 [8] and life-simulation game Spore [7].

We investigate here the feasibility of applying evolutionary techniques to develop competitive AI-bots to play commercial video games. The utility of this is two-fold, i.e. to enable intelligent agents to compete against human players in 1-player modes of games, and to act as avatars for players when they are not able to play themselves (e.g. as temporary substitutes in multiplayer games). The application of genetic programming has seen positive results in the fields such as robotic games [10] and board games like Chess [6], as well as racing track generation [13]. It has also been used to evolve human-competitive artificial players for Quake3 [12], and real-time strategy games [5]. By investigating the feasibility of

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 100–110, 2010.

applying an evolutionary approach with behaviours trees to develop a competitive automated AI player for a commercial game, we hope to further exemplify it as a viable means of AI development that may be adopted by the video games industry. We demonstrate this by evolving a competitive automated player for the game DEFCON, which is able to beat the hand-designed AI-bot written by the programmers at Introversion Software Ltd. more than 50% of the time over a large number of games. We provide relevant background reading in section 2, outlining DEFCON and behaviour trees. Next, we describe how evolution is applied to behaviour trees in section 3, and provide a description of the different fitness functions we employed in section 4. The experiments conducted and results obtained are in section 5, and we conclude and propose areas where future work may be applicable in section 6.

2 Background

2.1 DEFCON

DEFCON[1] is a commercial multiplayer real-time strategy game that allows players to take the roles of the military commanders of one of six possible world territories. Players are given a set of units and structures at their disposal, and have to manage these resources and inflict the greater amount of damage against opposing players. The game is split into 5 discrete time intervals (DEFCON5 to DEFCON1), and each dictate the movements and manipulation of units that are allowed. There are 7 available **territories** in each game, with each controlled by up to 1 party. A **party** represents the player, either human or AI-controlled, each allocated a fixed quantity of units that it may place and make use of throughout the course of the game. Playing the game involves strategic planning and decision making in coordinating all these units and winning by attaining the highest score, calculated via various scoring modes. We used the default scoring mode: 2 points awarded for every million of the opponent's population killed and a point penalty for every million people belonging to the player lost.

Several implementations of automated players exist for DEFCON, and we refer to these as **AI-bots**. For example, the default bot that comes with DEFCON is a deterministic, finite-state-machine driven bot. It consists of a set of 5 states and transits from one state to the next in sequence. Upon reaching the final state, it remains in it until the end of the game. In 2007, an AI-bot was developed by Baumgarten [2] using a combination of case-based reasoning, decision tree algorithms and hierarchical planning. For the case-based reasoning system, high-level strategy plans for matches were automatically created by querying a case base of recorded matches and building a plan as a decision tree, which classified each case's plan dictating the placement of fleets and units, and the sequence of movements and attacks. The starting territories of each player were used as a similarity measure for retrieving a case. The results, plans, and structure information were retained as a case in the case base at the end of a match,

[1] The official DEFCON website is here: http://www.introversion.co.uk/defcon

to enable an iterative learning process. Furthermore, the low-level behaviour of units, such as precisely timed bomber attacks and ship manoeuvres within fleets were added to improve the tactical strength of the AI-bot. As a result, in certain configurations, these low level modifications were able to influence the game outcome significantly, resulting in a victory for Baumgarten's AI-bot over the Introversion implementation in roughly 7 out of 10 matches on average.

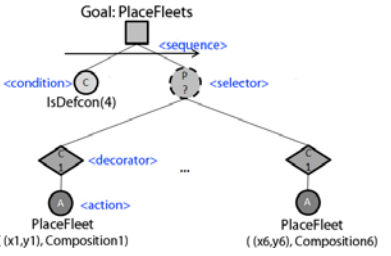


Fig. 1. Behaviour Tree: PlaceFleets

Both AI-bots covered above were implemented using the source-code of DEFCON, which gave it access to game state information that was required for various calculations. The implementation of the AI-bot described here made use of the an application programming interface (API)[2] that allows the control of a player in DEFCON by an external AI-bot, which can retrieve and invoke method calls on DEFCON, providing a way for developers to build AI-bots without having to work with the source-code of DEFCON directly. We avoided dictating low-level strategic decisions in our implementation but still successfully managed to evolve the behaviour trees to produce a competitive AI-bot with little human intervention. In a DEFCON competition competing API-designed AI-bots held at the Computational Intelligence and Games conference in Milan in August 2009, our AI-bot emerged victorious (although it was a rather hollow victory, as we were the only entrant to the inaugural competition!).

2.2 Behaviour Trees

A traditional approach to developing AI-controlled players for games has been to use Finite State Machines (FSMs). The problem with FSMs is that as the AI-bot grows in complexity, the number of states and transitions between them grows exponentially with the size of the game, making it difficult to manage. Even though Hierarchical Finite State Machines (HSFMs) overcome this, reusability is often a problem. Behaviour trees provide a simple, scalable and modular solution to embody complex AI behaviours. Each tree is goal-oriented, i.e. associated with a distinct, high-level goal which it attempts to achieve. These trees can be linked together with one another, allowing the implementation of complex behaviours by first defining smaller, sub-behaviours. Behaviour trees are constructed from 2 types of constructs. Firstly, primitive constructs form the leaves of the tree, and define low level actions which describe the overall behaviour. They are classified into 2 types, **actions**, which execute methods on the game, and **conditions**, which query the state of the game. Secondly, composite constructs can be used to group such primitive constructs to perform a higher-level function. the 3 main types of composites are **sequences**, **selectors** and **decorators**.

[2] API and documentation available at: http://www.doc.ic.ac.uk/~rb1006/projects:api

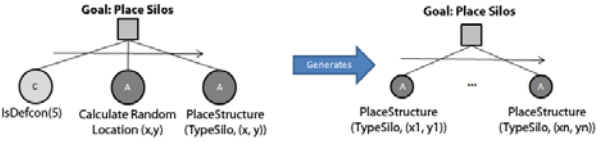


Fig. 2. Using A hand-crafted tree to generate a random behaviour tree

As an example, consider Figure 1 which shows a behaviour tree, with its nodes labelled to identify the constructs used. At the root, the tree has a **high-level goal** of placing a fleet. The **sequence** node dictates that, in order to achieve this goal, it has to first achieve the **sub-goal** of checking that the game is in DEFCON4 using a **condition**, followed by another composite sub-goal to place the fleet within the game. Thus, all child nodes must succeed for a **sequence** to succeed. The sub-goal is dictated by a **selector**, which will succeed as soon as one of its child nodes succeeds. Each child task begins with a **decorator**, which in this case acts as a counter to ensure that its child **action** node, used to place a fleet of a specified *composition* at location *(x,y)*, is only executed once. We read the tree as one that achieves the goal of placing a fleet by first checking that it is the correct point in the game to do so. Then, it selects one (x, y, composition) combination at random and executes it in the game. If that combination fails, it will try the next set combination until it has exhausted all of its options.

3 Evolving Behaviour Trees

We used behaviour trees to hand-craft an initial AI-bot to demonstrate that we could encode the basic abilities that a player would be able to perform. The AI-bot was given sufficient functionality to execute the basic actions to play DEFCON. However, its decision of when and whether to apply the actions were performed randomly. Ultimately, we planned to evolve the behaviour trees of this random AI-bot by playing games against the default Introversion AI-bot and afterwards, extract the best performing behaviour trees in different areas before combining them to produce a competitive AI-bot overall.

3.1 Randomly Generating Behaviour Trees

To produce the original set of behaviour trees, we adopted Bryson's Behaviour Oriented Design approach [3,4]. We define a high-level goal for the AI-bot before subsequently breaking it down into smaller sub-tasks that would form the basis of the required behaviour trees. This iterative process identifies building blocks of which to define more complex behaviours upon. Figure 2 shows how the hand-crafted tree on the left was used to produce a new tree on the right for thepurpose of placing silo units randomly. The left tree checks whether it is the appropriate DEFCON level, selects a coordinate at random, and places the silo at that randomly chosen location. The resultant tree capable of placing silos at those

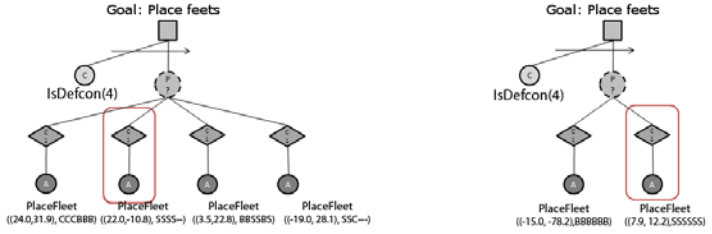


Fig. 3. Two behaviour trees with branches selected for recombination

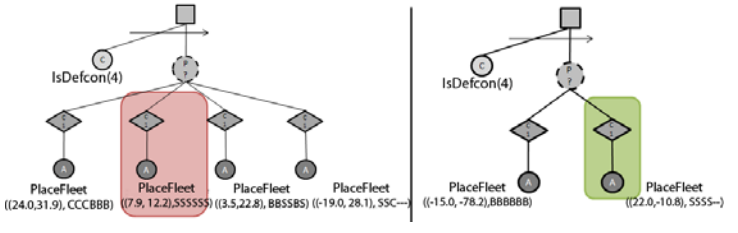


Fig. 4. The offspring from the recombination between the two parent behaviour trees

locations is shown on the right. We vary the AI-bot's behaviour (i.e. the number and positions of silos placed) by choosing which of its branches to attach or remove. We later made use of evolution to make these choices. We continued with this approach to produce other behaviour trees that performed random decisions with regards to other aspects of playing the game.

3.2 Applying Genetic Operators

Trees are structures which genetic operators are naturally applicable to [9], with crossovers on branches and mutation on nodes. Figure 3 shows two behaviour trees which have the same goal of placing fleet units. Using the left behaviour tree as a reference, what occurs is a `sequence` that first checks if the current DEFCON level is 4, and then proceeds to the priority selector if it is so. Looking at the left most `action` node, it will attempt to place a fleet composed of 3 carriers and 3 battleships at longitude 24.0 and latitude 31.9. The parent counter `decorator` ensures that the action is only executed once.

Crossovers are applied to randomly selected portions of the trees. Figure 4 shows the resulting offspring from recombination.Instead of placing a second fleet at (22.0, -10.8) with a composition of 4 submarines, it now places a second fleet at (7.90, 12.2) with a composition of 6 submarines. Random mutation can be used to increase genetic diversity. In Figure 5, the green portion shows how incremental mutation might occur to the left behaviour tree of figure 4, resulting in a different topology. Instead of placing 4 fleet units, the AI-bot now places 5 fleet units. The location and fleet compositions used for the new branch (highlighted green) were generated randomly during mutation. The red portion shows how a point mutation might occur. Since behaviour trees are not strongly-typed

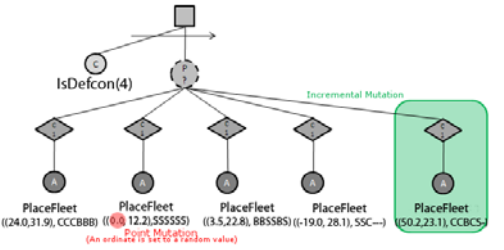


Fig. 5. Two types of mutations occurring in one of the behaviour trees

for recombination, inferior offspring trees may result (i.e. placing units which the AI-bot doesn't possess, or in illegal positions.) These trees would presumably be naturally selected against as the system evolves. In section 6, we mention ways to extend this approach to produce richer and more descriptive behaviour trees.

4 Fitness Functions

As mentioned previously, we evolved behaviour trees for individual behaviours, and combined the best performing trees into an overall AI-bot control mechanism. For the 4 behaviours, we used these fitness functions to drive the search:

• **Defence Potential of Silo Positions.** Silos are ground installations which play a significant role in both attack and defence. In both cases, a silo's effectiveness is affected not only by its absolute position, but also its position relative to the other silos. We chose to focus on the defensive aspect of the silos by measuring their **defence potential** – the total number of air units that were successfully destroyed for a given game.

• **Uncovered Enemy Units by Radars.** While radars do not have a direct influence on a player's attack or defensive abilities, they provide an indirect means of support by uncovering enemy units. This allows the AI-bot to then react appropriately to the situation, i.e. by sending fighters to enemy sea units or by shooting down missiles earlier. The coverage of the radars is determined by their positions, and we used the number of enemy sea units uncovered before the start of DEFCON1 as a fitness measure for evolving radar placement positions.

• **Fleet Movement & Composition.** The fitness of the fleet movement and composition in a game was calculated by evaluating the number of enemy buildings uncovered, the number of enemy buildings destroyed and the number of enemy units destroyed by the AI-bot. For each of these, a score was calculated and the average of the 3 scores taken as an overall fitness for the behaviour tree.

• **Timing of Attacks.** We used the difference between the final end-game scores as an indicator of how well the AI-bot performed for timing of attacks. A larger difference indicated a convincing win whereas a smaller difference would mean a narrower victory. We fixed the bounds for the maximum and minimum difference to +250 and -250 respectively (with these values found empirically through some initial experimentation). The fitness was calculated using the function:

$$fitness = \frac{difference_{score} - (-250)}{250 - (-250)} = \frac{(score(AIbot) - score(Enemy)) + 250}{500}$$

5 Experiments and Results

5.1 Experimental Setup

We chose the following game options for our experiments. Firstly, the starting territories for our AI-bot and the enemy were fixed as Africa and South America respectively. Secondly, the default game scoring was used. Four main experiments were performed, each evolving a set of AI-bots with the aim of improving the population AI-bot's performance as per the respective fitness functions described above. Each population contained 100 individuals, and each experiment was evolved between 80 to 250 generations. We employed a fitness proportionate selection method to choose pairs for recombination and set the mutation rate to 5%. These parameters were chosen after performing several initial experiments. Naturally, there is a vast array of other parameter settings we could have experimented with, but given the time constraints imposed by the number of experiments and running time of each game, we had to decide upon the values empirically. Via four evolutionary sessions, we evolved behaviour trees for:

1. Placement of silos to maximise defence against enemy air units
2. Placement of radar stations to maximise the number of enemy units detected throughout the course of the game
3. The placement, composition and movement of the player's fleet units to maximise their overall attack capabilities
4. The timings of the attacks for 4 types of attach, namely, submarine attacks using mid-range missiles, carrier bomber attacks using short range missiles, air base bomber attacks using short range missiles and silo attacks using long range missiles.

5.2 Distribution

Ultimately, the fitness functions rely on playing a game to completion. Unfortunately, with 4 experiments, each running for about 100 generations with 100 individuals in a population would require 40,000 game runs. With each game taking approximately 90 seconds to complete, a total time of 3.6 million seconds (∼41 days) of continuous processing would be required for the project. To bring the time-frame down to ∼2 days per experiment, we distributed the running of DEFCON games over 20 computers, connected together via a local area network.

5.3 Results

For silo placements, Figure 6(a) shows that the mean fitness of the population increased over the generations. The mean number of missiles destroyed increases

from around 70 to almost 100, and similarly, the mean number of bombers destroyed increases from about 18 to about 34. However, the mean number of fighters destroyed remained at around 18 across the generations, which we believe is due to the silo placement locations evolving further away and out of enemy fighter scout range over the generations. For radar placements, we observe a similar increase in mean fitness. Figure 6(b) shows the mean number of detected enemy sea units increasing from around 6 to 10, with the mean fitness of the top 10% of the population even reaching 21.5 (∼90% of detectable enemies when excluding submarines) at generation 80. For fleet placement, composition and movement, Figure 6(c) shows an increase in the AI-bots' mean fitness as the AI-bot evolved over the generations, with the mean fitness increasing from about 8 to 26. Similarly, when evolving the attack timings for the AI-bot, Figure 6(d) shows an increase in the average number of games won. Initially having on average 4 wins in a population of 100 AI-bots, the number reached 47 at Generation 65. The average number of wins appears to be plateauing, which might indicate a need to continue the evolution over more generations.

We constructed an AI-bot with a controller that used the best trees evolved for each of the four behaviours. It was set to play 200 games against the default Introversion AI-bot. The difference between the scores obtained by both the AI-bot and the opponent was used as an indicator of how well the AI-bot had performed, which we term as the margin of victory. Prior to performing the evolution, the AI-bot which consisted of behaviour trees which performed random movements and choices (Section 3.1) managed to win around 3% of the

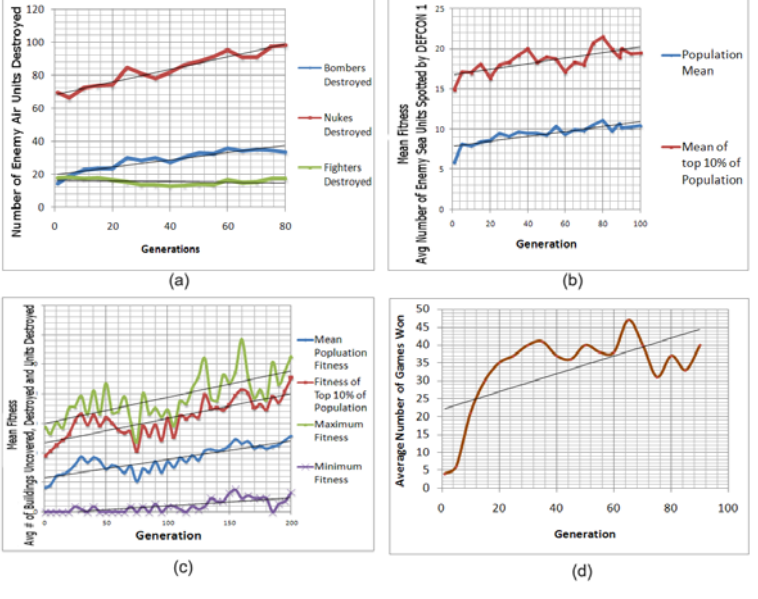


Fig. 6. Mean fitnesses over generations for each behaviour. (a) silo placement (b) radar placement (c) fleet coordination (d) attack timings.

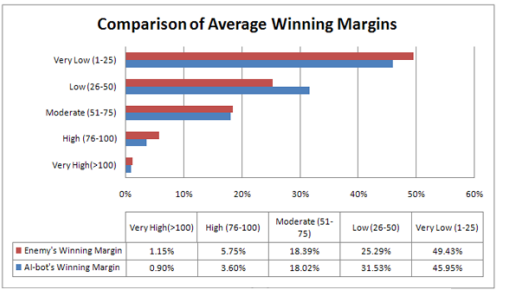


Fig. 7. Summary of the Performance of the Evolved AI-bot

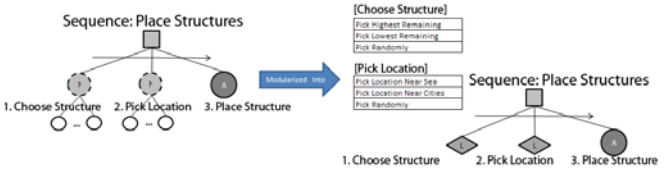


Fig. 8. Generating different topologies for random trees

time out of the 200 games. We ran the evolved AI-bot which beat the default Introversion AI-bot 55% of the time in 200 games. Figure 7 shows the distribution of the margins of victory in these matches. We note that in a large number of games, our AI-bot only lost to the opponent by a very low margin, indicating that the the number of wins by our AI-bot could have been larger. The opponent managed to beat our AI-bot by a moderate to a very high margin fairly often, indicating more convincing victories. Using a binomial test at the 4% significance level with a 50% winning probability, we were unable to reject the null hypothesis that both AI-bots were equal in abilities.

6 Conclusions and Future Work

By evolving behaviour trees for individual behaviours and combining the trees into an overall AI-bot, we have been able to produce a competitive player that was capable of beating the original, hand-coded DEFCON AI-bot more than 50% of the time. This hints at the possibility that such an approach is indeed feasible in the development of automated players for commercial games. Speculating on the effect of further experimentation, we refer to the graphs in Figure 6. Although we have seen improvements after approximately 100 generations, we notice that the mean fitness seems to have reached a plateau, which might indicate that performing the evolution for a greater number of generations may not show significant improvements in mean fitness. This raises the question of whether evolutionary techniques need to be supplemented with other techniques in automating AI-bot design, and if so, which techniques should be investigated.

Evolving against a single opponent could have caused over-fitting. An improvement would be to perform experiments against other AI-bots or human players. Also, in the event that no training AI-bot was present, it raises the question of whether co-evolution [12] could have been applied. DEFCON is a non-deterministic game, especially when involving human players. We did not consider all possible permutations of starting locations for both the player and the opponent. Africa and South America are within close proximity, so other starting locations would increase the distance between players and might require different strategies. We also picked 4 tasks to concentrate on, but there are other game-play aspects such as the coordination of air units that could have been investigated. Our implementation only considered the transition between two stages, from being defensive to launching an attack, which had to occur in that order. It would be interesting to see the application of the evolution of sub-trees using `lookup` decorators to allow the AI-bot to exhibit complex behaviours and adaptive game play styles to match opponents, resulting in more descriptive behaviour trees. Figure 8 shows how the use of `decorators` can be used to generate behaviour trees which may differ in topology.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments.

References

1. Bauckhage, C., Thurau, C.: Exploiting the fascination: Video games in machine learning research and education. In: Proceedings of the 2nd International Workshop in Computer Game Design and Technology (2004)
2. Baumgarten, R., Colton, S., Morris, M.: Combining AI Methods for Learning Bots in a Real-Time Strategy Game. Int. J. of Computer Games Tech. (2009)
3. Bryson, J.: Action selection and individuation in agent based modelling. In: Proceedings of the Argonne National Laboratories Agent Conference (2003)
4. Bryson, J.: The behavior-oriented design of modular agent intelligence. In: Kowalczyk, R., Müller, J.P., Tianfield, H., Unland, R. (eds.) NODe-WS 2002. LNCS (LNAI), vol. 2592, pp. 61–76. Springer, Heidelberg (2003)
5. Hagelbäck, J., Johansson, S.: Using multi-agent potential fields in real-time strategy games. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-agent Systems, vol. 2 (2008)
6. 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)
7. Hecker, C., McHugh, L., Argenton, M., Dyckhoff, M.: Three Approaches to Halo-style Behavior Tree AI. In: Games Developer Conference, Audio Talk (2007)
8. Isla, D.: Managing complexity in the Halo 2 AI system. In: Proceedings of the Game Developers Conference (2005)

9. Langdon, W.: Size fair and homologous tree genetic programming crossovers. Genetic programming and evolvable machines 1(1/2), 95–119 (2000)
10. Luke, S.: Genetic programming produced competitive soccer softbot teams for RoboCup. In: Proceedings of the 3rd Annual Conference of Genetic Programming (1998)
11. Orkin, J.: Three states and a plan: the AI of FEAR. In: Proceedings of the Game Developers Conference (2006)
12. Priesterjahn, S., Kramer, O., Weimer, A., Goebels, A.: Evolution of human-competitive agents in modern computer games. In: Proceedings of the IEEE Congress on Evolutionary Computation (2006)
13. Togelius, J., De Nardi, R., Lucas, S.: Towards automatic personalised content creation for racing games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2007)

Evolving 3D Buildings for the Prototype Video Game Subversion

Andrew Martin, Andrew Lim, Simon Colton, and Cameron Browne

Computational Creativity Group
Department of Computing, Imperial College London
www.doc.ic.ac.uk/ccg

Abstract. We investigate user-guided evolution for the development of virtual 3D building structures for the prototype (commercial) game Subversion, which is being developed by Introversion Software Ltd. Buildings are described in a custom plain-text markup language that can be parsed by Subversion's procedural generation engine, which renders the 3D models on-screen. The building descriptions are amenable to random generation, crossover and mutation, which enabled us to implement and test a user-driven evolutionary approach to building generation. We performed some fundamental experimentation with ten participants to determine how visually similar child buildings are to their parents, when generated in differing ways. We hope to demonstrate the potential of user-guided evolution for content generation in games in general, as such tools require very little training, time or effort to be employed effectively.

1 Introduction

Creating content for a game world is a notoriously time-consuming process, largely due to the complexities involved in 3D modelling. Of all of the environments that can be created in a 3D world, one of the most difficult to realise is a city, and yet they are becoming increasingly common in modern games. Significant amounts of time and money are required to create such environments, and even for larger companies, this can represent a sizeable financial risk. Furthermore, it is often necessary to make significant concessions in order to achieve a result within an appropriate time frame, such as repeatedly reusing the same assets or settling for an unrealistically small game environment. Subversion is an early game prototype by Introversion Software Ltd. (www.introversion.co.uk), which will be set in an automatically generated city. The aim here was to create a tool that would allow a player to rapidly design buildings for the Subversion game world, to increase their engagement with the game and to make the 3D environment richer and more diverse.

As described in section 2, buildings in Subversion are represented in a custom structural markup language. This representation is amenable to random generation, crossover and mutation, as described in section 3. Hence, we were able to implement a user-directed building creation interface which users/players can employ to evolve a set of satisfactory and unique building models from an initial

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 111–120, 2010.

generation of stock (or randomly generated) models. If this interface is to find its way into the Subversion release, various experiments relating to user satisfaction have to be carried out. In section 4, we describe the most fundamental such experiment. This tested whether the evolutionary operators lead to useful changes, i.e., the user can make satisfying progress using the tool, where child buildings look enough like their parents for there to be a purpose to choosing candidates for evolution, but are not so similar that progress is too slow. As described in section 5, other approaches could be used to solve this problem, such as providing the user with intuitive modelling tools similar to those seen in the popular game Spore. However, the main advantage of the evolutionary approach is that the user/designer need not have a clear image of their desired result in mind before proceeding. Instead, the user is presented with visual cues for the duration of the design process, and simply has to visually assess each building model presented to him/her. This also allows for a satisfyingly minimalist interface, far removed from the rather intimidating interfaces normally found in 3D modelling tools.

2 Building Generation

The Subversion procedural content engine is capable of generating and displaying 3D models of building represented as data files that describe their key features as sets of structural commands. The language was custom designed for the description of buildings and the resulting data files are in human readable form.

2.1 Command Language

The command language essentially defines each building as a stack of three-dimensional objects, each described as a two-dimensional shape that is vertically extruded. The two-dimensional base shapes include circles, rectangles and squares, and can be subject to various simple operations such as translation, rotation and union. Base shapes may also be contracted, and it is also possible to offset the base of a shape from its initial position using a three-dimensional vector, which can be used in conjunction with a contraction to create slanted edges. Examples of the command code used to describe two buildings are shown in figure 1a, with an illustrative example given below. The Subversion interpreter parses the building description files into internal tree representations, which can then be used to rapidly generate 3D models of the buildings as required. An advantage of internally representing objects in tree form is that this facilitates their manipulation by evolutionary techniques, as described in section 3.

2.2 An Illustrative Example

Figure 2 portrays a pair of building files, where the first, `CirclePart`, has another building (`SquarePart`) embedded on top of it. The internal data structure that is used to represent the buildings is a node grid which consists primarily of a list of vertices in addition to fields for vertical extrusions and height. The `Creation`

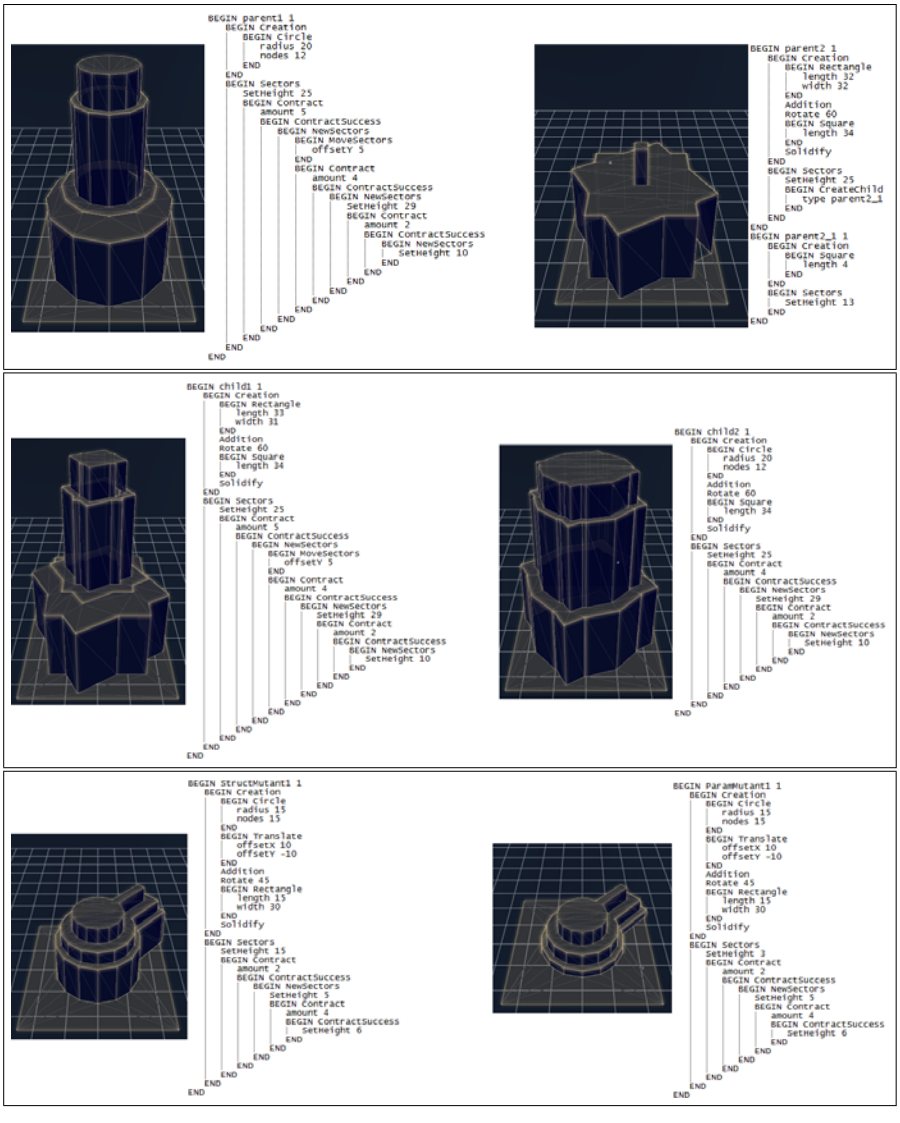


Fig. 1. (a) Two example buildings and their Subversion command code descriptions. (b) Two children produced by crossing over the parents above. (c) A structural mutant [left] and a parametric mutant [right].

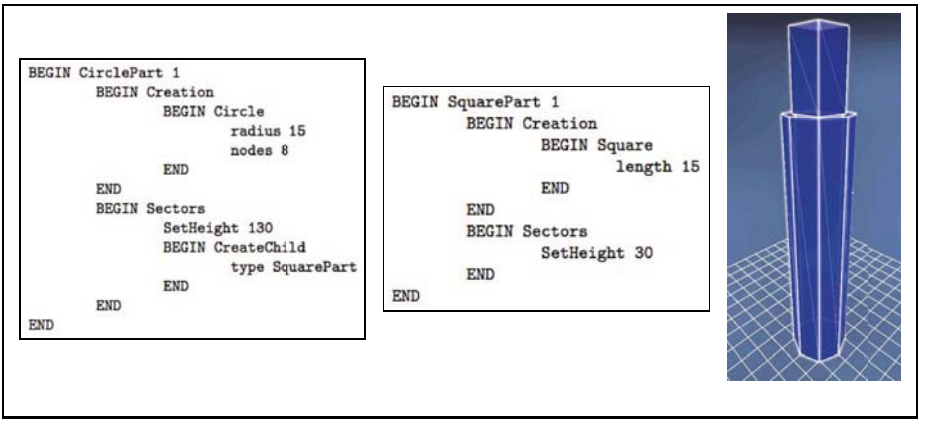


Fig. 2. A simple building represented as two building files, and the resulting node grid

section of each file in figure 2 expresses what the base shape for the building is going to be. It allows for basic shapes like circles, rectangles and squares as well as more complex shapes specified by vertex lists, which can also be subject to various simple operations like rotation and union. The `Sectors` parts of building files define transformations to be applied using the shape defined in `Creation`, and they also allow for the embedding of other buildings by the `CreateChild` command. The transformations include: rotation (which is applied to the entire base shape as opposed to the per part rotation done in `Creation`), scaling and contraction (which is the result of moving all the edges along their inner normal at a constant rate, as shown in figure 2 by the protruding tower).

When parsed, the building file is split into two separate sections called `Creation` and `Sectors`. As mentioned previously, when the generator is called by the game engine, it uses the information in the file to generate a 3D node grid, represented with a vertex list, height and child information. In our example, the node grid generated by `CirclePart` after Creation is:

$$\text{vertexlist} = \{(15, 0, 0), (10.6, 0, 10.6), (0, 0, 15), (-10.6, 0, 10.6), (-15, 0, 0), (-10.6, 0, -10.6), (0, 0, -15), (10.6, 0, -10.6)\}$$
$$\text{height} = 0, \text{child} = \text{null}.$$

In this example, immediately after the base has been generated, the `Sectors` commands will be run and for the `CirclePart` node grid, the vertexlist would remain unchanged, but the height would become 30 and the child field would point to a new node grid, that eventually becomes the node grid for the `SquarePart`:

$$\text{vertexlist} = \{(7.5, 0, 7.5),(-7.5, 0, 7.5),(-7.5, 0, -7.5),(7.5, 0, -7.5)\}$$
$$\text{height} = 30 \text{ child} = \text{null}$$

After a building has been generated into a node grid, the grid is triangulated and passed to the renderer. The buildings themselves are then simply represented as

an array of triangles. The colour and alpha values of the triangles to be rendered are stored elsewhere in the engine, where the RGBA values of everything that is to be rendered inside the Subversion city are stored.

3 Evolving Buildings

Our building evolution tool maintains a population of building models and operates using a standard evolutionary approach. The population is initially seeded with the randomly generated building descriptions, and those buildings which are selected by the user constitute the breeding pool used to create the next generation. If the breeding pool is empty when the user requests a new generation, then more random buildings are produced for the next generation. If not, then random pairs of parents are selected from the breeding pool, crossed over to create children and then mutated, based on the likelihoods specified through the user interface (see subsection 3.3 below). This process is repeated until enough children have been created to fill the next generation.

3.1 Crossover

Crossover is performed as per standard Genetic Programming techniques. Each branch of the parent trees is assigned a tag denoting that branch's role and what other branches it is compatible with. For example, branches tagged `Circle` and `Square` both denote two-dimensional base shapes, hence are functionally equivalent within the language and therefore compatible. Similarly, a branch tagged as a command to define extrusion length may be crossed over with other similarly tagged branches. Each command within the subset of the language that we use has a corresponding tag.

After the trees are tagged, both parents are cloned and a branch is randomly selected from one of the clones. All branches in the other clone with matching tags are then located and one is randomly selected; these two compatible branches are then swapped. This process of swapping branches may then be performed multiple times, depending on the user-specified crossover strength. Once the swapping is completed, one of the modified clones will be deleted, and the other saved as the child. Two examples of crossover can be found in figure 1b. For example, the building depicted in figure 1b [left] is a child produced by single-swap crossover, taking buildings A and B depicted in figure 1a respectively as its two parents. The child has taken A as its base, and has had its `Creation` branch swapped for the `Creation` branch of B. The building depicted in figure 1b (right) is a child produced by four-swap crossover between A and B. The child has taken A as its base again, but this time the `Circle` subtree was swapped for the `Rectangle` subtree of B, and then the entire `Creation` subtree was swapped for the (now modified) `Creation` subtree of B, bringing the `Circle` subtree back into A. Similarly, the entire `Contraction` subtree was first swapped out for the `CreateChild` subtree of B, but then the `CreateChild` subtree was swapped back out for a new `Contraction` subtree beneath the original Contraction subtree. As we can see from figures 1a and 1b, the two children are clearly different from both parents, but have inherited aspects of both.

3.2 Mutation

After crossover, the following two kinds of mutation may be applied to the resulting offspring, depending on the user-specified strengths.

• **Structural Mutations.** These apply to the structure of the tree itself, so that some subtrees are completely replaced with randomly generated subtrees (which are functionally equivalent, as explained in section 3.1). In order to structurally mutate a tree, another tree is randomly generated with the constraint that it must contain the same number of sections as the first tree. Swapping is then performed between the trees as per crossover. As before, the strength of the structural mutation is defined as the number of swaps performed between the randomly generated tree and the original tree. Figure 11c [left] shows a structural mutation of a building. A random tree was generated containing a `Contract` subtree, which was then swapped into the original building in place of the `CreateChild` subtree.

• **Parametric Mutations.** These apply to those commands with numerical parameters, such as the sizes of shapes, rotations, translations, extrusions and contractions. The process is straightforward: numerical parameters are selected at random from the tree and replaced with new values randomly generated within each parameter's specified range. For example, a value specifying the degree of rotation might be mutated to any value between 0 and 360. The strength of the parametric mutation dictates the number of parameters that will be mutated. Figure 11c [right] shows a typical parametric mutation. Note that significant changes may result from the parametric mutation of buildings with complex Creation subtrees, but as we see from the experiments below, weak parametric mutations will probably not have a significant effect.

3.3 User Interface

Upon starting a session with the building evolution tool, the user is shown an initial generation of randomly generated data files, each describing a building, for which the corresponding 3D models are generated and displayed as a grid of tiles. The user can control a camera in virtual 3D space to view the buildings from any desired angle; every building is viewed from the same angle to avoid the tedium of manipulating a separate camera for each. Clicking on a tile expands that building to fill the screen, and clicking again reverts the interface back to the tile view. Each tile has a button to indicate its status as an individual of interest. The interface also provides a number of sliders to control (a) the structural mutation strength and likelihood (b) the parametric mutation strength and likelihood; and (c) the crossover strength. When the user has finished selecting buildings of interest and configuring the sliders, they click on the *Next Generation* button to create a new generation of buildings from those selected. The tile grid is then cleared and repopulated with the new generation; previous generations can be revisited using tabs along the top of the grid.

4 Experimental Survey and Results

With user-driven evolutionary systems such as those found in evolutionary art, the first question to be asked is whether the evolution of artefacts provides a satisfying experience. To be satisfying, the user must feel like (a) their choices make a difference, i.e., the children in each subsequent generation bear an adequate resemblance to the parents that the user chose and (b) evolution is not proceeding too slowly, i.e., the children in subsequent generations should not look too much like their parents. Especially when evolving complex objects such as building designs which are marked up, we should not take such a satisfying search for granted, as it is possible that the objects are so complex that even minor changes at genome level make huge changes at the artefact level.

For a fundamental experiment to address this question, and to test whether the different generation techniques produce children as expected, ten participants were asked to complete a survey where two parent buildings and 16 offspring buildings were presented for twenty pairs of parents. Figure 3 shows an example of the questionnaire sheet. We see that the participant is asked to rate each child as being (a) too similar (b) too dissimilar or (c) neither: okay to their parents. Designs marked as 'too different' often included those that were too outlandish or physically infeasible to make pleasing buildings, while those that showed trivial variation from either parent and therefore offered little genetic diversity were often marked as too similar. The nature of the way in which the 16 children were generated was varied across the twenty questionnaire sheets. In particular, the following six generation methods were exhibited, with the sheets taken from the 1st, 6th and 14th populations, to produce 18 sheets:

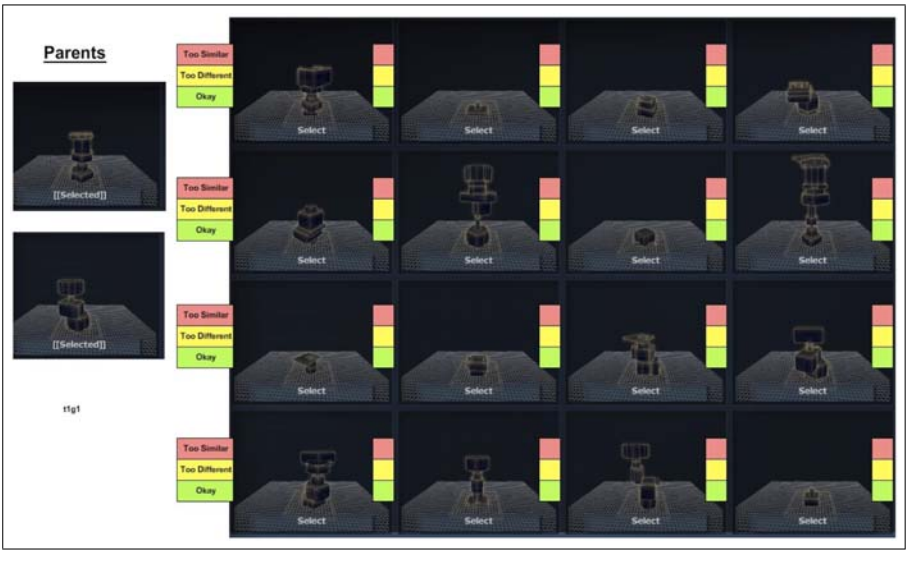


Fig. 3. Example questionnaire sheet

Table 1. Building designs by their construction method, with evaluation percentages and average evaluation percentages per construction method indicated

Construction Method	Generation Number	Participant Evaluation (%) Too similar	Too different	OK
Strong crossover	1	1.9	52.8	45.3
	6	9.4	43.1	47.5
	14	4.4	47.5	48.1
	average	5.2	47.8	47.0
Weak crossover	1	47.2	5.0	47.8
	6	49.4	4.4	46.2
	14	35.6	14.4	50.0
	average	44.1	7.9	48.0
Strong structural	1	1.3	51.9	46.8
	6	6.3	43.7	50.0
	14	4.4	70.0	25.6
	average	4.0	55.2	40.8
Weak structural	1	51.6	12.6	35.8
	6	11.9	29.4	58.7
	14	17.5	8.7	73.8
	average	37.0	16.9	56.1
Strong parametric	1	1.3	56.9	41.8
	6	3.8	39.4	56.8
	14	0.6	49.4	50.0
	average	1.9	48.6	49.5
Weak parametric	1	23.3	25.8	50.9
	6	28.7	11.3	60.0
	14	35.6	1.3	63.1
	average	29.2	12.8	58.0
Random	1	0.0	56.2	43.8
	14	1.3	78.7	20.0
	average	0.7	67.4	31.9

1. Strong crossover (20 swaps per child).
2. Weak crossover (5 swaps per child).
3. Strong structural mutation (20 structural mutations per child).
4. Weak structural mutation (5 structural mutations per child).
5. Strong parametric mutation (20 parametric mutations per child).
6. Weak parametric mutation (5 parametric mutations per child).

Two sheets with randomly generated children were also used as controls.

4.1 Results

The results of the surveys are shown in table 1. The values shown indicate the percentages of those designs marked too similar, too different and OK respectively. The results were largely as to be expected. In particular, it is clear that the random generations bear little resemblance to the supposed parents (on average only 0.7% of the children were deemed too similar, with 67.4% of the children deemed too different). As expected, the randomly generated control set performed worse than the controlled parameter sets, with children being

marked as OK much less often on average than any other generation method. It is interesting to note, however, that 31.9% of randomly generated children were deemed to be neither too similar nor too dissimilar (OK), which suggests that some random introduction of building designs might be appreciated by the user, and might enable them to drive the search in new directions. In overview, it is encouraging that the non-random generation methods achieve a hit rate of around 50% for OK children, although this could be improved upon.

As expected, we see that weak versions of crossover, structural mutation and parametric mutation all produced children which were assessed far more often as too similar to the parents than the strong versions of these generation methods. Perhaps less expected is the observation that the weak form of each parameter setting led to more child designs being marked by the user as OK than the corresponding strong form, and we plan further experimentation to determine why this might be. Furthermore, we can also see that the weak structural mutation and weak parametric mutation settings led to more children being marked as OK than any other settings, although this improvement is not marked. The only obvious correlation with the generation number is exhibited by the weak parametric generation method, where the percentage of children marked as too different clearly decreases as the generation number increases. This can be explained by the size of the building design tree growing as the session progresses, and because weak parametric mutation has little effect on larger designs.

5 Conclusions and Future Work

A good survey of procedural techniques for city generation is supplied in [3]. The most well known example of procedural building generation is Pascal Mueller et. al's work [4], which forms part of the city generation procedures [5], which has resulted in the commercial CityEngine package (www.procedural.com). Mueller's approach is different from the one presented here, because it employs a shape grammar rather than a markup language, and is discussed primarily in the context of heavily automated generation. While shape grammars have been evolved, for instance in [9], we do not know of any application of evolutionary techniques to Mueller's approach. A genetic algorithms approach to conceptual building design is presented in [8], where the emphasis is on decision support systems to assist designers. Addressing fundamental questions of whether the evolutionary techniques involved in evolutionary arts projects (in particular, image filter evolution) are better than random generation is the topic of [1], where the authors asked 30 participants to interact with a very similar interface (without the 3D elements) to the one described here. They showed that intelligent generation of image filters is preferred by users up to statistical significance, and the evolutionary techniques outperformed image retrieval and database techniques. We plan to compare and contrast our approach with other evolutionary approaches to content generation for games, such as in [2].

We have demonstrated that a user-directed evolutionary process can be employed for generating custom building designs for game content in a city world

for a commercial video game. We have further shown that the search process the user undertakes is satisfying because it neither produces too many overly similar nor overly dissimilar offspring buildings. As suggested by an anonymous reviewer, we need to investigate how disruptive the crossover mechanism is that we employ, and consider experimenting with homologous crossover methods [6] [7]. Moreover, while our preliminary experiments have served an experimental design purpose here, we need to gather more data in order to draw statistically significant conclusions about the nature of user interaction with the system. In future, we plan to develop a more sophisticated user interface for the evolution and selection process, by adding more constraints to both the command language and the generation process. This will hopefully reduce the number of outlandish and infeasible shapes that are invariably rejected by the user, which could be further enhanced by the automated detection of such degenerate cases. Another useful addition to the tool would be the inclusion of constraints for the generation of themed content for situations in which particular building types are required. Examples of such themes might include 'corporate skyscraper', 'residential house', and so on. Also, more sophisticated experiments comparing combinations of parameter settings rather than each parameter setting in isolation could reveal further insights into harmonious parameter settings for the purpose of automated evolution.

Acknowledgements

We would like to thank the anonymous reviewers for their very useful comments. This work was supported by a Technology Strategy Board grant.

References

1. Colton, S., Gow, J., Torres, P., Cairns, P.: Experiments in objet trouvé browsing. In: Proceedings of the 1st Int. Joint Conference on Computational Creativity (2010)
2. Hastings, E., Guha, K., Stanley, K.: Evolving content in the galactic arms race video game. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2009)
3. Kelly, G., McCabe, H.: A survey of procedural techniques for city generation. Institute of Technology Blanchardstown Journal 14 (2006)
4. Mueller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modelling of buildings. ACM Transactions on Graphics 25(3), 614–623 (2006)
5. Parish, Y., Mueller, P.: Procedural modelling of cities. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (2001)
6. Park, J., Park, J., Lee, C., Han, M.: Robust and efficient genetic crossover operator: Homologous recombination. In: Proc. Int. Joint Conf. on Neural Networks (1993)
7. Poli, R., Stephens, C., Nucleares, C., Wright, A., Rowe, J.: On the search biases of homologous crossover in linear genetic programming and variable-length genetic algorithms. In: Proc. Genetic and Evolutionary Computation Conference (2002)
8. Rafiq, Y., Mathews, J., Bullock, G.: Conceptual building design-evolutionary approach. Journal of Computing in Civil Engineering 17(3), 150–158 (2003)
9. Saunders, R., Grace, K.: Extending context free to teach interactive evolutionary design systems. In: Proceedings of the EvoMUSART workshop (2009)

Finding Better Solutions to the Mastermind Puzzle Using Evolutionary Algorithms

Juan J. Merelo-Guervós[2] and Thomas Philip Runarsson[1]

[1] School of Engineering and Natural Sciences, University of Iceland
tpr@hi.is
[2] Department of Architecture and Computer Technology, ETSIIT, University of Granada, Spain
jmerelo@geneura.ugr.es

Abstract. The art of solving the Mastermind puzzle was initiated by Donald Knuth and is already more than thirty years old; despite that, it still receives much attention in operational research and computer games journals, not to mention the nature-inspired stochastic algorithm literature. In this paper we revisit the application of evolutionary algorithms to solving it and trying some recently-found results to an evolutionary algorithm. The most parts heuristic is used to select guesses found by the evolutionary algorithms in an attempt to find solutions that are closer to those found by exhaustive search algorithms, but at the same time, possibly have better scaling properties when the size of the puzzle increases.

1 Introduction

Mastermind in its current version is a board game that was introduced by the telecommunications expert Mordecai Merowitz [12] and sold to the company Invicta Plastics, who renamed it to its actual name; in fact, Mastermind is a version of a traditional puzzle called *bulls and cows* that dates back to the Middle Ages. In any case, Mastermind is a puzzle (rather than a game) in which two persons, the *codemaker* and *codebreaker* try to outsmart each other in the following way:

- The codemaker sets a length ℓ combination of κ symbols. In the classical version, $\ell = 4$ and $\kappa = 6$, and color pegs are used as symbols over a board with rows of $\ell = 4$ holes; however, in this paper we will use uppercase letters starting with A instead of colours.
- The codebreaker then tries to guess this secret code by producing a combination.
- The codemaker gives a response consisting on the number of symbols guessed in the right position (usually represented as black pegs) and the number of symbols in an incorrect position(usually represented as white pegs).
- The codebreaker then, using that information as a hint, produces a new combination until the secret code is found.

For instance, a game could go like this: The codemaker sets the secret code *ABBC*, and the rest of the game is shown in Table 1.

Different variations of the game include giving information on which position has been guessed correctly, avoiding repeated symbols in the secret combination (*bulls and*

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 121–130, 2010.

Table 1. Progress in a Mastermind game that tries to guess the secret combination *ABBC*. The player here is not particularly clueful, playing a fourth combination that is not *consistent* with the first one, not coinciding, for instance, in two positions and one color (corresponding to the 2 black/1 white response given by the codemaker) with it. The rest of the combinations would effectively be *consistent*; for instance, ABBD coincides in two places (first A and third B) and one "color" (B) with the first, and two positions (A and B in the first and second position) with the second combination.

Combination	*Response*
AABB	2 black, 1 white
ABFE	2 black
ABBD	3 black
BBBE	2 black
ABBC	4 black

cows is actually this way), or allowing the codemaker to change the code during the game (but only if this does not make responses made so far false).

In any case, the codebreaker is allowed to make a maximum number of combinations (usually fifteen, or more for larger values of κ and ℓ), and the score corresponds to the number of combinations needed to find the secret code; after repeating the game a number of times with codemaker and codebreaker changing sides, the one with the lower score wins.

Since Mastermind is asymmetric, in the sense that the position of one of the players after setting the secret code is almost completely passive, and limited to give hints as a response to the guesses of the codebreaker, it is rather a puzzle than a game, since the codebreaker is not really matching his skills against the codemaker, but facing a problem that must be solved with the help of hints, the implication being that playing Mastermind is more similar to solving a Sudoku than to a game of chess; thus, the solution to Mastermind, unless in a very particular situation (always playing with an opponent who has a particular bias for choosing codes, or maybe playing the dynamic code version), is a search problem with constraints.

What makes this problem interesting is its relation to other, generally called *oracle* problems such as circuit and program testing, differential cryptanalysis and other puzzle games (these similarities were reviewed in our previous paper [8]) is the fact that it has been proved to be NP-complete [11,5] and that there are several open issues, namely, what is the lowest average number of guesses you can achieve, how to minimize the number of evaluations needed to find them (and thus the run-time of the algorithm), and obviously, how it scales when increasing κ and ℓ. This paper will concentrate on the first issue.

This NP completeness implies that it is difficult to find algorithms that solve the problem in a reasonable amount of time, and that is why initial papers [8,2] introduced stochastic evolutionary and simulated annealing algorithms that solved the Mastermind puzzle in the general case, finding solutions in a reasonable amount of time that scaled roughly logarithmically with problem size. The strategy followed to play the game was optimal in the sense that is was guaranteed to find a solution after a finite number of combinations; however, there was no additional selection on the combination played other than the fact that it was consistent with the responses given so far.

In this paper, after reviewing how the state of the art in solving this puzzle has evolved in the last few years and showing how different evolutionary algorithms fare against each other, we try to apply some kind of selection to consistent solutions found by the evolutionary algorithm, so that the combination played is most likely to shorten the search time. In this we follow a path initiated by Berghman et al. [1], but taking into account our own results [10] on the number of combinations that are needed to obtain a reasonable result. The main idea driving this line of research is to try and obtain results that are comparable with the exhaustive search methods, but without the need to *see* all possible combinations at the same time. This will allow to create a method that can work, in principle, for any problem size, and scale reasonably unlike exhaustive search whose scaling is exponential in time *and* in memory.

The rest of the paper is organized as follows: next we establish terminology and examine the state of the art; then the new evolutionary algorithms we introduce in this paper are presented in section 3 and the experimental results in 4; finally, conclusions are drawn in the closing section 5.

2 State of the Art

Before presenting the state of the art, a few definitions are needed. We will use the term *response* for the return code of the codemaker to a played combination, c_{played}. A response is therefore a function of the combination, c_{played} and the secret combination c_{secret}, let the response be denoted by $h(c_{played}, c_{secret})$. A combination c is *consistent with* c_{played} iff

$$h(c_{played}, c_{secret}) = h(c_{played}, c) \tag{1}$$

that is, if the combination has as many black and white pins with respect to the played combination as the played combination with respect to the secret combination. Furthermore, a combination is *consistent* iff

$$h(c_i, c) = h(c_i, c_{secret}) \text{ for } i = 1..n \tag{2}$$

where n is the number of combinations, c_i, played so far; that is, c is *consistent with* all guesses made so far. A combination that is consistent is a candidate solution. The concept of consistent combination will be important for characterizing different approaches to the game of Mastermind.

One of the earliest strategies, by Knuth [6], is perhaps the most intuitive for Mastermind. In this strategy the player selects the guess that reduces the number of remaining consistent guesses and the opponent the return code leading to the maximum number of guesses. Using a complete minimax search Knuth shows that a maximum of 5 guesses are needed to solve the game using this strategy. This type of strategy is still the most widely used today: most algorithms for Mastermind start by searching for a consistent combination to play.

In some cases once a single consistent guess is found it is immediately played, in which case the object is to find a consistent guess as fast as possible. For example, in [8] an evolutionary algorithm is described for this purpose. These strategies are fast and do not need to examine a big part of the space. Playing a consistent combinations

eventually produces a number of guesses that uniquely determine the code. However, the maximum, and average, number of combinations needed is usually high. Hence, some bias must be introduced in the way combinations are searched. If not, the guesses will be no better than a purely random approach, as solutions found (and played) are a random sample of the space of consistent guesses.

The alternative to discovering a single consistent guess is to collect a set of consistent guesses and select among them the best alternative. For this a number of heuristics have been developed over the years. Typically these heuristics require all consistent guesses to be first found. The algorithms then use some kind of search over the space of consistent combinations, so that only the guess that extracts the most information from the secret code is issued, or else the one that reduces as much as possible the set of remaining consistent combinations. However, this is obviously not known in advance. To each combination corresponds a partition of the rest of the space, according to their match (the number of blacks and white pegs that would be the response when matched with each other). Let us consider the first combination: if the combination considered is AABB, there will be 256 combinations whose response will be 0b, 0w (those with other colors), 256 with 0b, 1w (those with either an A or a B), etc. Some partitions may also be empty, or contain a single element (4b, 0w will contain just AABB, obviously). For a more exhaustive explanation see [7]. Each combination is thus characterized by the features of these partitions: the number of non-empty ones, the average number of combinations in them, the maximum, and other characteristics one may think of.

The path leading to the most successful strategies to date include using the *worst case*, *expected case*, *entropy* [9,3] and *most parts* [7] strategies. The *entropy* strategy selects the guess with the highest entropy. The entropy is computed as follows: for each possible response i for a particular consistent guess, the number of remaining consistent guesses is found. The ratio of reduction in the number of guesses is also the *a priori* probability, p_i, of the secret code being in the corresponding partition. The entropy is then computed as $\sum_{i=1}^{n} p_i \log_2(1/p_i)$, where $\log_2(1/p_i)$ is the information in bit(s) per partition, and can be used to select the next combination to play in Mastermind [9]. The *worst case* is a one-ply version of Knuth's approach, but Irving [4] suggested using the *expected case* rather than the worst case. Kooi [7] noted, however, that the size of the partitions is irrelevant and that rather the number of non empty partitions created, n, was important. This strategy is called *most parts*. The strategies above require one-ply look-ahead and either determining the size of resulting partitions and/or the number of them. Computing the number of them is, however, faster than determining their size. For this reason the *most parts* strategy has a computational advantage.

Following a tip in one of the papers that tackled Mastermind, recently Berghman et al. [1] proposed an evolutionary algorithm which finds a number of consistent guesses and then uses a strategy to select which one of these should be played. The strategy they apply is similar the *expected size* strategy. However, it differs in some fundamental ways. In their approach each consistent guess is assumed to be the secret in turn and each guess played against every different secret. The return codes are then used to compute the size of the set of remaining consistent guesses in the set. An average is then taken over the size of these sets. Here, the key difference between the *expected size* method is that only a subset of all possible consistent guesses is used and some return

codes may not be considered or considered more frequently than once, which might lead to a bias in the result. Indeed they remark that their approach is computationally intensive which leads them to reduce the size of this subset further.

The heuristic strategies described above use some form of look-ahead which is computationally expensive. If no look-ahead is used to guide the search a guess is selected purely at *random*, and any other way of ranking solutions might find a solution that is slightly better than random, but no more. However, it has been shown [10] that in order to get the benefit of using look-ahead methods, an exhaustive set of all consistent combinations is not needed; a 10% fraction is sufficient in order to find solutions that are statistically indistinguishable from the best solutions found. This is the path explored in this paper.

3 Description of the Method

Essentially, the method described is an hybrid between an evolutionary algorithm and the exhaustive partition-based methods described above. Instead of using exhaustive search to find a set of consistent combinations, which then are compared to see the way they partition that set, we use evolutionary algorithms to find a set of consistent combinations and compute then the partitions they yield. The size of the set is fixed according to our previous results [10], which show that a set of size 20 is enough to obtain results that are statistically indistinguishable from using the whole set of consistent combinations. By using this approach we are assuming that the set of consistent combinations found by the evolutionary algorithm is random; since evolutionary search introduces a bias in search space sampling, this need not be the case.

The evolutionary algorithms used are similar to the Estimation of Distribution Algorithm (EDA) that was also shown in our previous paper. The fitness function is similar to the one proposed by Berghman et al. [1], except for the term proportional to the number of positions; that is,

$$f(c_{guess}) = \sum_{i=1}^{n} |h(c_i, c_{guess}) - h(c_i, c_{secret})| \tag{3}$$

which is the number of black and white peg changes needed to make the current combination c_{guess} consistent; this number is computed via the absolute difference between the number of black and white pegs h the combination c_i has had with respect to the secret code c_{secret} (which we know, since we have already played it) and what c_{guess} obtains when matched with c_i. For instance, if the played combination *ABBB* has obtained as result $2w, 1b$ and our c_{guess} *CCBA* gets $1w, 1b$ with respect to it, this difference will be $(2-1)w + (1-1)b = 1$. This operation is repeated over all the combinations c_i that have been played. In the previous paper [10] a *local entropy* was used to bias search, but in this paper we found this was harmful, so we have left it out. However, the first combination was fixed to *ACBA*, same as before.

While in [10] the EDA presented proceeded until a consistent solution was found, and then played the one with the highest local entropy (since they were ranked by local entropy), in this work we will proceed in the following way:

- Continue evolutionary search until at least 20 consistent solutions are found.
- If a set of 20 solutions is not found, continue until the number of consistent solutions does not change for three generations. This low number was chosen to avoid stagnation of the population.
- If at least a single consistent solution is not found after 15 generations, reset the population substituting it by a random one. Again, this was a number considered high enough to imply stagnation of search and at the same time give the algorithm a chance to find solutions.

If any of the conditions above hold (either a big enough set is found, or the algorithm has not found any more solutions for 3 generations), a set of consistent solutions is obtained; for each solution in this set, the way they partition the rest of the set is computed. Then, the solution which leads to the maximum number of non-zero partitions (*most parts*) is computed, in a similar way as proposed by [7] and differently from Berghman [1], which use some form of expected size strategy. As shown by [10], *most parts* and the entropy method are the best, however, *most parts* is faster to compute. We wanted also to find out whether this strategy obtained better solutions than the plain vanilla evolutionary algorithm.

Several evolutionary algorithms were also tested. A rank-based steady state algorithm obtained generally good results, but there were occasions (around one in several thousands) where it could not find it even with restarts, so it was eliminated. Finally we settled for an EDA and a canonical genetic algorithm. In both cases letter strings where used for representing the solutions; that means that when mutation is used, a string mutation (change a letter for another in the alphabet) is used. Source code and all parameter files used for these experiments are available as GPL'ed code from `http://sl.ugr.es/alg_mm/`. The parameters used for both algorithms are shown in table 2. No exhaustive tweaking of them has been made.

Results obtained with these algorithms are shown in the next section.

Table 2. Parameter values in the evolutionary algorithms used in this paper

Parameter	EDA	CGA
Operators	N/A	Mutation, 1-point crossover
Population	300	256
Replacement Rate	50%	40%

4 Experimental Results

In the same way as was done by ourselves [10] and Berghman [1], we have tested the algorithms on every possible combination, but instead of doing it 3 times, we have run the algorithm ten times for each combination from $AAAA$ to $FFFF$ to obtain a higher statistical certainty. The whole 10K runs last several hours on an Intel quad-core computer. The number of combinations obtained by both methods are shown in table 3 and compared with others, results for *most parts* and *entropy* are taken from our former paper, and Berghman's from his paper.

Table 3. Comparison of results obtained for different Mastermind-solving algorithms. Entropy and *Most parts* are taken from [10]; μ is the maximum size of the set of consistent solutions used to find the solution. The EDA algorithm shown is the one that uses local entropy in fitness, as explained in the text. Berghman uses different sizes, and we include only the smallest and biggest; when the value has not been published, an empty slot is shown. The last two lines show the results obtained by the algorithms presented in this paper.

Strategy	*min*	*mean*	*median*	*max*	*st.dev.*	*max guesses*
Entropy $\mu = \infty$	4.383	4.408	4.408	4.424	0.012	6
Most parts $\mu = \infty$	4.383	4.410	4.412	4.430	0.013	7
Entropy $\mu = 20$	4.438	4.468	4.476	4.483	0.016	7
Most parts $\mu = 20$	4.429	4.454	4.454	4.477	0.016	7
EDA $\mu = 1$	4.524	4.571	4.580	4.600	0.026	7
Berghman $\mu = 30$	-	4.430	-	-	-	7
Berghman $\mu = 60$	-	4.390	-	-	-	-
CGA $\mu = 20$	4.402	4.434	4.433	4.471	0.018	7
EDA $\mu = 20$	4.419	4.445	4.448	4.467	0.017	7

The results obtained by these new methods, CGA and EDA, are competitive with the exhaustive search methods with $\mu = 20$, that is, they are all statistically equivalent. All statistical tests were performed using the Wilcoxon rank sum test for equal medians at a 0:05 significance level. When $\mu = 30$ they become also statistically equivalent to using the whole set of consistent guesses, i.e. $\mu = \infty$, resulting in a method comparative to exhaustive search. The results also show that sampling of the combination set, as performed by the evolutionary algorithm, is actually random, since it behaves in the same way as shown by the random sampling algorithm. In fact, this should be expected, at least for EDA, since the results obtained by a plain-vanilla, play-as-find the solution go method, were statistically indistinguishable from a random algorithm. Besides, EDA and CGA need roughly the same number of evaluations to find those solutions, although, on average, EDA is better, since the maximum number of evaluations needed by the CGA is six times bigger than the worst result obtained by the EDA.

On the other hand, and all things being equal[1] the results appear very similar to those obtained by Berghman with the smallest set size μ. We cannot say if they are statistically indistinguishable, but it is quite likely that they actually are. We say this, since the *expected size* strategy is worse than the *most parts* and *entropy* methods [10]. The running time, for an interpreted (not compiled) code which has not been optimized in any way, is around one second per game, which is comparable to those published by them[2].

How do these algorithms actually work to obtain these results? Since the initial pool has a good amount of combinations, they usually include a big enough pool of consistent combinations, sufficient for the second play and on occasions also the third. The

[1] That is, assuming the average has been computed in the same way as we did.

[2] But please note that machines are quite different too. That only means that running times are on the same order of magnitude.

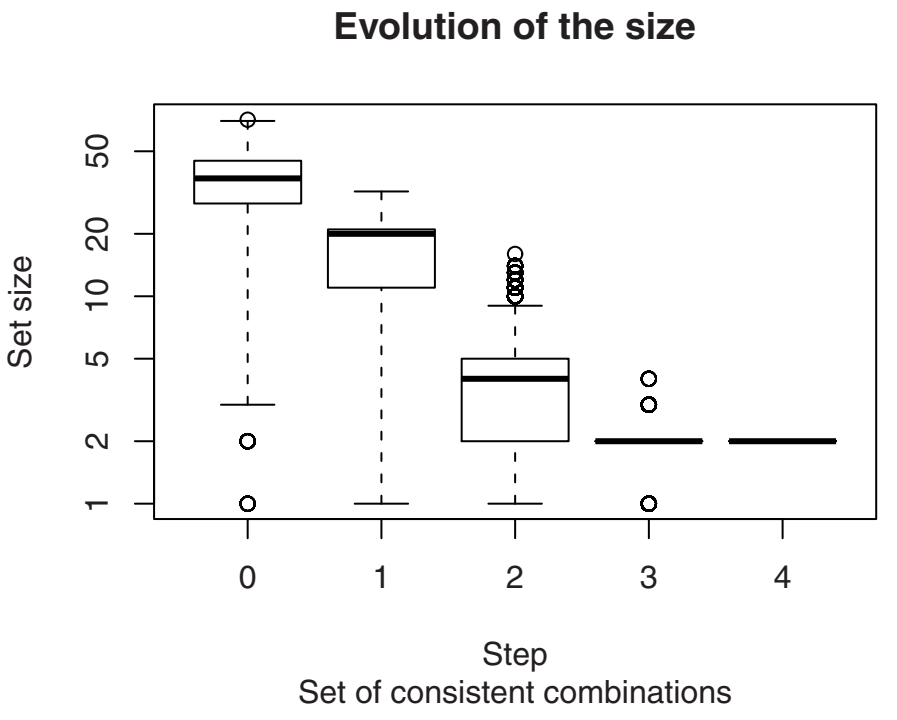


Fig. 1. Boxplot of the evolution of set size with the game, horizontal line in the box indicates the average, the box upper and lower side the quartiles, and dots outside it outliers; the x axis (labeled *Step* represents the number of combinations played after the first one, with the y axes representing the set size; thus, at $x = 0$ the set size after the first combination has been played has been represented. The y axis has been scaled logarithmically

remaining draws, notably including the secret code, must be found by the evolutionary algorithm. In fact, the average number of elements in the set of consistent solutions evolves with combinations played as shown in figure 1.

In that figure, the first set has a size close to 50, and the second is pegged at approximately 20, indicating that the evolutionary algorithm has probably intervened to find a few consistent solutions. For the rest of the combinations it is quite unusual to find a set of that size, at least with the constraints we have set; eventually the 4th and 5th draws are drawn from a single set. The figure also shows an idea of how set sizes should scale for bigger sizes of the Mastermind problem; however, this hunch should be proved statistically.

5 Conclusion and Future Work

In this paper we have introduced a new evolutionary algorithm that tries to find the solution to the game of Mastermind, in as few games as possible, by finding a set of

possible solutions and ranking them according to the most part heuristic. This approach is as good as other systems that use exhaustive heuristic search, comparable to other methods that represent the state of the art in evolutionary solution of this game, and does not add too much overhead in terms of computing or memory, using time and memory roughly in the same order of magnitude as the simple evolutionary algorithm.

This also proves that exhaustive heuristic search strategies can be successfully translated to evolutionary algorithms, provided that the size of the sample is tested in advance. This might prove a bit more difficult as the size increases, but a systematic exploration of the problem in different sizes might give us a clue about this. Our hunch right now is that it will scale as the logarithm of the problem size, which would be a boon for using evolutionary algorithms for bigger sizes.

This is probably an avenue we will pursue in the future: compute the size set of consistent guesses needed to find the correct solution in a minimal number of draws, and apply it to evolutionary algorithms of different shapes. One of these could be to find a way of including the score of a consistent solution in the evolutionary algorithm by using this as a fitness, and taking into account its consistency only as a constraint, not as the actual fitness. This will prove increasingly necessary as the size of the consistent solution sets increases with problem space size.

The space of EAs has not been explored exhaustively in this paper either. Although it is rather clear that different parameter settings will not have an influence on the quality of play, it remains to be seen how they will impinge on the number of evaluations needed to find the consistent combinations. Different combinations of operators and operator rates will have to be tested. And as the complexity of the system goes up with the combination length and number of colors, distributed approaches will have to be tested, which could have an influence not only on the running time, but also on the number of evaluations needed to find it. These will be explored in the future.

Acknowledgements

This paper has been funded in part by the Spanish MICYT projects NoHNES (Spanish Ministerio de Educación y Ciencia - TIN2007-68083) and TIN2008-06491-C04-01 and the Junta de Andalucía P06-TIC-02025 and P07-TIC-03044.

References

1. Berghman, L., Goossens, D., Leus, R.: Efficient solutions for Mastermind using genetic algorithms. Computers and Operations Research 36(6), 1880–1885 (2009), http://www.scopus.com/inward/record.url?eid=2-s2.0-56549123376
2. Bernier, J.L., Herráiz, C.I., Merelo-Guervós, J.J., Olmeda, S., Prieto, A.: Solving mastermind using GAs and simulated annealing: a case of dynamic constraint optimization. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 554–563. Springer, Heidelberg (1996), http://citeseer.nj.nec.com/context/1245314/0

3. Bestavros, A., Belal, A.: Mastermind, a game of diagnosis strategies. Bulletin of the Faculty of Engineering, Alexandria University (December 1986), http://citeseer.ist.psu.edu/bestavros86mastermind.html, available from http://www.cs.bu.edu/fac/best/res/papers/alybull86.ps
4. Irving, R.W.: Towards an optimum mastermind strategy. Journal of Recreational Mathematics 11(2), 81–87 (1978–1979)
5. Kendall, G., Parkes, A., Spoerer, K.: A survey of NP-complete puzzles. ICGA Journal 31(1), 13–34 (2008), http://www.scopus.com/inward/record.url?eid=2-s2.0-42949163946
6. Knuth, D.E.: The computer as Master Mind. J. Recreational Mathematics 9(1), 1–6 (1976–1977)
7. Kooi, B.: Yet another Mastermind strategy. ICGA Journal 28(1), 13–20 (2005), http://www.scopus.com/inward/record.url?eid=2-s2.0-33646756877
8. 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), http://www.sciencedirect.com/science/article/B6W86-4FH0D6P-1/2/40a99afa8e9c7734baae340abecc113a, http://dx.doi.org/10.1016/j.asoc.2004.09.003
9. Neuwirth, E.: Some strategies for Mastermind. Zeitschrift fur Operations Research. Serie B 26(8), B257–B278 (1982)
10. Runarsson, T.P., Merelo, J.J.: Adapting heuristic Mastermind strategies to evolutionary algorithms. In: NICSO 2010 Proceedings. LNCS. Springer, Heidelberg (2010) (to be published) ArXiV: http://arxiv.org/abs/0912.2415v1
11. Stuckman, J., Zhang, G.Q.: Mastermind is NP-complete. CoRR abs/cs/0512049 (2005)
12. Wikipedia: Mastermind (board game) — Wikipedia, The Free Encyclopedia (2009), http://en.wikipedia.org/w/index.php?title=Mastermind_board_game&oldid=317686771 (Online; accessed 9-October-2009)

Towards a Generic Framework for Automated Video Game Level Creation

Nathan Sorenson and Philippe Pasquier

School of Interactive Arts and Technology,
Simon Fraser University Surrey, 250 -13450 102 Avenue, Surrey, BC
{nds6,pasquier}@sfu.ca

Abstract. This paper presents a generative system for the automatic creation of video game levels. Our approach is novel in that it allows high-level design goals to be expressed in a top-down manner, while existing bottom-up techniques do not. We use the FI-2Pop genetic algorithm as a natural way to express both constraints and optimization goals for potential level designs. We develop a genetic encoding technique specific to level design, which proves to be extremely flexible. Example levels are generated for two different genres of game, demonstrating the system's broad applicability.

Keywords: video games, level design, procedural content, genetic algorithms.

1 Introduction

Procedural content creation, which is the algorithmic generation of game assets normally created by artists, is becoming increasingly important in the games industry [1]. Not only does this automation provide a way to produce much more content than would be possible in the typical game development process, but it could also allow for game environments to be adapted to individual players, providing a more personalized and entertaining experience.

Current level generation techniques [2,3] tend to be bottom-up, rule-based systems which iterate over a set of production rules to construct an environment. These rules must be carefully crafted to create playable levels, and design goals are restricted to the emergent behaviour of the rule set. Furthermore, these techniques are extremely idiosyncratic; no standard toolbox or library exists for game content generation, and every game requires the construction of a specialized algorithm. These factors result in level generation systems that can take more effort to construct than the levels themselves [1].

We describe an approach that avoids these issues by allowing designers to specify the desired properties of the generated level, instead of requiring them to specify the details of how levels are assembled. This is done using a Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm [4], which is an evolutionary technique that proves to be well suited to the present domain. Level designers specify a set of constraints, which determine the basic requirements for a level to be considered playable. The "infeasible population" consists solely of levels which do not yet satisfy all these constraints, and these individuals are evolved towards minimizing the number of constraints violated. When individuals are found to satisfy all the constraints, they are moved to the "feasible

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 131–140, 2010.

population" where they are subjected to a fitness function that rewards levels based on any criteria specified by the level designers. Essentially, this population is for generating levels that are not only playable, but also fun.

A simple, yet expressive genetic encoding for levels is presented, based on the specification of simple *design elements*. These design elements are the building blocks used by the GA to construct the level. We argue for the flexibility of this encoding by providing encodings for two different types of games: one is the 2D platformer *Super Mario Bros* [5], and the other is an exploration-adventure game similar to *The Legend of Zelda* [6]. With these encodings, we are able to evolve levels that satisfy a number of game-specific constraints and, furthermore, are fun. To measure how enjoyable a level is, we use a generic model of challenge-based fun described in previous work [7], and we apply this model to the generated levels of both types of games. Early results indicate that our approach is indeed useful and likely to be widely applicable.

2 Previous Work

Generative systems are often used to as a means to theoretically analyze the nature of games, as opposed to simply providing a way to create more game content. Togelius and Schmidhuber use GAs to generate novel game designs [8] that are subsequently evaluated with a neural net to test how fun they are, relying on the hypothesis that machine-learnable games are more entertaining. Similarly, Yannakakis and Hallam [9] explore the relationship between difficulty and fun by generating computer opponents of varying degrees of skill. Smith et. al. [10] share our goal of automatically generating game levels for 2D platformers; however, they use a generative grammar, which is a bottom-up, rules-based approach to constructing game levels which is tied to this specific genre. Pedersen et al. [11] generate levels for 2D platformers with the goal of modeling player experience. Their model suggests how player behaviour and player fun are related, however it does not provide any details on how specific level configurations (at the local level) influence player enjoyment and it is unclear how this model would effectively inform an automated level generation process. As well, their generative technique is restricted to the genre of 2D platformers.

In general, these applications of generative systems employ specific algorithms that are tailored to address precisely defined research questions in a single domain, whereas our goal is to present a widely applicable framework, usable in a variety of contexts.

3 Implementation

3.1 Feasible-Infeasible Two-Population Genetic Algorithm

Optimization is clearly an aspect of level design; game levels are constructed in order to maximize the amount of fun a player experiences. However, levels generally consist of a spatial arrangement of rooms, platforms, doors, and other components, and if these elements do not align properly, the entire level can become unplayable. For these reasons, the process of game design must be seen as both and optimization problem and a constraint satisfaction problem. GAs are effective in solving high-dimensional optimization problems, but are, in general, ineffective when tasked with solving constraint

satisfaction problems [12]. Gradual, incremental improvement becomes impossible if there are too many constraints on the feasibility of the levels, and this causes the evolutionary algorithm's performance to suffer. Conversely, the nuanced and complex notion of fun does not lend itself readily to the simple finite-domain predicates required by most constraint solvers.

Kimbrough et al. [4] present the Feasible-Infeasible Two-Population genetic algorithm (FI-2Pop) as an effective way to address these problems. FI-2Pop maintains two separate populations, one containing only feasible solutions and the other containing only infeasible solutions. Any solution that satisfies each specified constraint is moved into the feasible population, and any solution that violates a constraint as a result of mutation or crossover is moved into the infeasible population. FI-2Pop therefore requires two different fitness functions: the first is a typical optimization function that drives incremental changes towards a global optimum; the second fitness function is specifically for generating individuals that satisfy a set of constraints. Kimbrough et al. suggest that even a simple operation, such as counting the number of constraints violated, serves as an effective way to guide the population towards feasibility. We essentially follow this approach by summing up a set of individual constraint functions, which are each responsible for measuring a specific kind of constraint violation.

The two populations exert evolutionary pressure on one another through frequent migration, and because infeasible individuals are not simply killed off, a degree of genetic diversity can be maintained. Because level design criteria generally consist of a number of hard constraints and a number of soft, high-level goals, we find the FI-2Pop GA to be well suited to our domain. Though a multitude of evolutionary techniques have been developed to address similar concerns [13], FI-2Pop is a technically straightforward and conceptually simple approach to handling heavily constrained optimization problems. The choice of this particular algorithm is not the central contribution of this work; it is important to emphasize that, with our generic genotype encoding, we will be able to employ a more sophisticated and efficient algorithm should it prove necessary.

3.2 Genetic Representation

Because we desire this system to maintain as much generality as possible, care must be taken in the design of the genotype. Our genotype is therefore based on the notion of a *design element* (DE). DEs are small, composable building blocks that represent logical units of levels. For example, the DE for a game like *Breakout* [14] would be an individual block. DEs can be parameterized; the *Breakout* block DE could have parameters for both x and y coordinates, as well as for its physical properties. DEs do not have to be atomic, and can be specified at any level of abstraction. For example, a DE might be defined to create a star pattern of blocks, which would be parameterized by the size of the star. Essentially, DEs constitute a one-to-one genotype-to-phenotype mapping, as they literally describe a physical level element. They are therefore simple to define, and their behaviour is easy to predict.

The genotype representation consists of a variable-size set of these design elements. To allow for the use of typical variable-point crossover, we must impose an order onto the set to treat it as a linear genotype. The DEs can be sorted by an arbitrary function of their parameters, but crossover is most effective when genes exhibit strong genetic

linkage [15]. It is therefore best to sort the genotype in such a way that will tend to keep mutually influential genes together, maximizing the probability that beneficial arrangements of genes are not disrupted by crossover. Because level design is largely spatial in nature, we sort based on the coordinate parameter of the design elements, with the expectation that mutually influential genes will predominately represent level elements that are in close proximity. The sorting decision is especially straight-forward in the case of games such as 2D platformers; because all level elements are principally arranged along the horizontal axis, we sort the genotype based on the x coordinate. However, the second example we provide demonstrates that our approach is not restricted to linear games and indeed works in two-dimensional situations as well.

Our mutation operator adjusts an arbitrary parameter of a random design element gene in a genotype. Continuous values, such as height or width, are displaced using a normal distribution with variance derived from their permissible range, and categorical values are simply given a new permissible value.

3.3 Fitness Function

To generate enjoyable levels, we employ a fitness function that is able to identify how fun a given level is. Certainly, the notion of fun is exceedingly complex and difficult to define precisely. However, as a starting point, we can identify aspects of fun that are more tractable for computational analysis. For example, a large number of theorists [16,17,18,19] identify the presence of an appropriate level of challenge as integral to nature of fun in games. This is particularly true for a skills-based action game such as *Super Mario Bros.* Broadly speaking, a player has the most fun when presented with challenges that are neither too easy nor too difficult. We currently use a generic model of player enjoyment that is not restricted to a particular game, and is discussed in more detail in [7]. The model does not require any genre-specific information, instead it relies only a simple challenge measurement, $c(t)$ to determine the perceived difficulty a player experiences at time t, and rewards levels for matching a desired challenge configuration. In other words, this model is used to characterize the amount of fun, f, that is acquired by a player throughout the course of a level. The model is summarized in Equation (1).

$$\frac{df}{dt} = m * c(t) \tag{1}$$

The variable m can take the value of $+1$ or -1, and represents two important states of the model at a given time. When $m = 1$, the amount of fun measured increases with the challenge measurement. However, when $m = -1$, challenge serves to reduce the amount of fun. We specify threshold values that determine when the value m changes. When the amount of challenge in a given time period has exceeded the upper threshold, m becomes negative. Conversely, if not enough challenge has been measured , as determined by the lower threshold, m becomes positive. This model, in practice, tends to reward level designs that interpose periods of high difficulty with segments of low difficulty, and even though the challenge metric and model of fun are rough approximations to reality, they have been constructed in a principled manner and appear to produce acceptable results. Devising and characterizing such fitness functions is certainly a difficult question and will continue to be a topic for future study. We must

emphasize, however, that our framework does not necessarily depend on any particular characterization of level quality. Indeed, any fitness function can be created to express the subjective design goals of the game developer.

4 Validation Results

4.1 Super Mario Bros.

Our first example of this genetic encoding is based on the original *Super Mario Bros.* (SMB) [5]. SMB is a 2D platformer game, in which levels consist of an arrangement of platforms and enemies. Inspecting existing levels from the original game, we identify a number of design elements occur frequently, which are shown in Figure 1:

1. $Block(x, y)$. This DE is a single block, parameterized by its x and y coordinate.
2. $Pipe(x, height, piranha)$. A pipe serves as both a platform and a possible container of a dangerous piranha plant.
3. $Hole(x, width)$. This specifies a hole of a given width in the ground plane.
4. $Staircase(x, height, direction)$. Staircases are common enough to warrant a dedicated DE. The direction specifies whether the stairs are ascending or descending.
5. $Platform(x, width, height)$. This specifies a raised platform of a given width and height.
6. $Enemy(x)$. This specifies an enemy at the given horizontal location.

In addition to the DEs, we provide the following constraint functions for the infeasible population:

1. $require\text{-}exactly(n, type)$. This function allows designers to specify the desired number of certain types of design elements to be present in individuals. As a penalty, it returns the absolute difference between the counted number of instances of $type$ and the desired amount n.

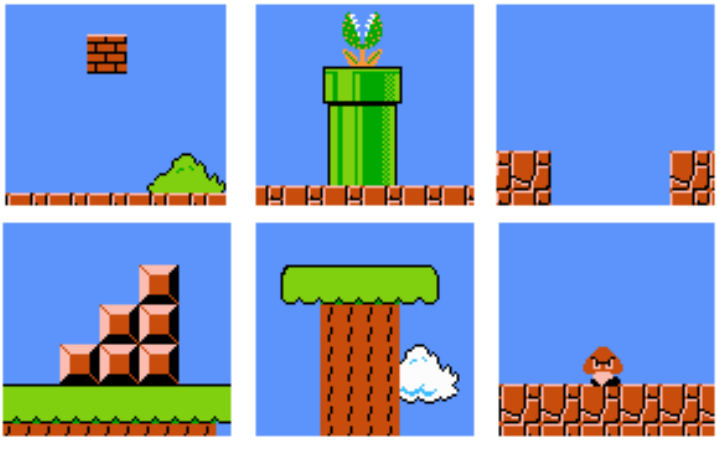


Fig. 1. The 6 DEs for SMB level design. Clockwise from top-left: block, pipe, hole, enemy, platform, and staircase DE.

2. $require\text{-}at\text{-}least(n, type)$. This function penalizes levels that contain less than n of a given $type$, returning 0 if $n \geq type$ and returning $type - n$ otherwise.
3. $require\text{-}at\text{-}most(n, type)$. This function penalizes levels that contain more than n of a given $type$, returning 0 if $n \leq type$ and returning $n - type$ otherwise.
4. $require\text{-}no\text{-}overlap(type_1, type_2, ...)$. This function states that the specified types are not to overlap in the phenotype. It is, therefore, only relevant for design elements that contain a notion of location and extent. In the present example, we specify that pipes, stairs, enemies, and holes should not overlap one another. As a penalty, the number of overlapping elements is returned.
5. $require\text{-}overlap(type_1, type_2)$. Though similar to function 4, this function specifies that $type_1$ must overlap $type_2$, though $type_2$ need not necessarily overlap $type_1$. We use this function to require that platforms must be positioned above holes. The number of $type_1$ elements that do not overlap with a $type_2$ element is returned.
6. $traversible()$. This function is to ensure that a player can successfully traverse the level, meaning that there are no jumps that are too high or too far for the player to reach. This is determined using a simple greedy search between level elements. The penalty is the number of elements from which there is no subsequent platform within a specified range, that is, the number of places a player could get stuck.

All the previous functions are specified such that a value of 0 reflects a satisfied constraint and a positive value denotes how severely a constraint is violated. Therefore, any individual level that is given a score of 0 by all of the above functions is considered a feasible solution and is moved into the feasible population for further optimization.The feasible population is evaluated using our generic model of challenge-based fun. We adapt this model to 2D platformers by providing a method for estimating challenge at any given point in a level. This is done by a function that returns a challenge value for each jump required between platforms, with difficult jumps being rated higher, and a set constant for each enemy in the level.

With no pressing concern for efficiency, we choose to set the mutation rate to 10% of individuals per generation and to generate the rest via crossover, using tournament selection of size 3. Finally, following the convention of Kimbrough [4], we limit the sizes of the infeasible and feasible populations to 50. Our stopping criterion is reached if the fitness of the levels does not improve for twenty generations. Figure 2 depicts some resulting levels.

A significant advantage of the evolutionary approach is the adaptability of the solution. For example, it is possible for an artist to hand-craft certain portions of a level, and have the GA automatically fill in the gaps according to the specified constraints. Consider the manually-created arrangement and the resulting evolved level in Figure 3. No extra functionality was needed to provide this behaviour; the genotype was simply hard-coded to always include the user-specified arrangement of DEs.

4.2 2D Adventure Game

A major disadvantage of typical generative systems is that they are restricted to a single application. Any improvements to the generative technique will benefit only that particular application. We claim that our evolutionary framework provides the ability to factor

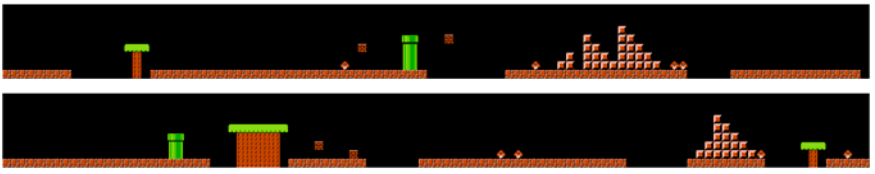


Fig. 2. Two different levels, created in 892 and 3119 generations, respectively. The number of staircases, platforms, and enemies are specified through constraints. On a mid-range dual-core laptop, the running time was for each was less than 30 minutes.

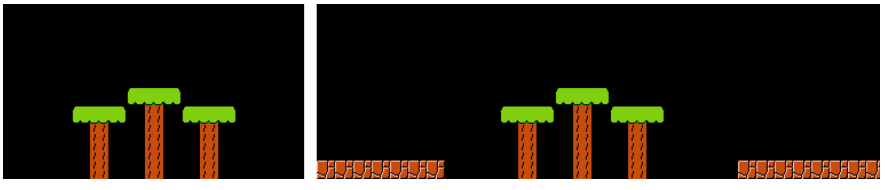


Fig. 3. An explicitly specified arrangement of platform DEs (left) is automatically wrapped in a surrounding hole (right)

out some of the generative logic from any game-specific context. For this to be the case, it must be relatively simple to express a variety of different game design goals without requiring fundamental changes to the underlying system. As a proof-of-concept example in support of this claim, we present a set of constraints for the evolution of levels for a simple top-down 2D adventure game similar to *The Legend of Zelda* [6].

The levels for this game will be constructed from three design elements:

1. $Hallway(x, y, length, direction)$. This codes for a hallway of a given length, whose direction can either be vertical or horizontal.
2. $Room(x, y, width, breadth)$. This creates a rectangular room of the specified size.
3. $Monster(x, y)$. This creates a monster at a given coordinate.

The genotype and the mutation and crossover operators are the same as in the previous example. Even though the coordinates of the design elements must now be expressed as (x, y) pairs instead of as a single x coordinate, we find that sorting by x to linearize the genotype produces acceptable results.

We specify two constraints for this simple game:

1. $connected(start, end)$. Returns 0 if there is a 4-connected path between the start and end points. Otherwise, penalizes levels for the minimum distance between the two areas reachable from the start and end points. In other words, levels that are far from being connected are penalized more than areas that are nearly connected.
2. $require\text{-}overlap(type_1, type_2)$. We use this constraint, introduced in the 2D platformer experiment, to ensure monsters are located in rooms or hallways. The number of monsters that do not overlap hallways or rooms is returned.

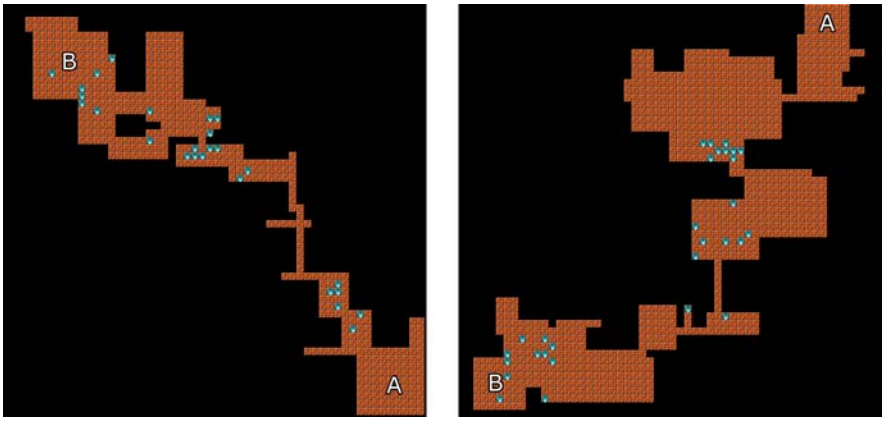


Fig. 4. A small and a large level, created in 1629 and 2330 generations, respectively. The starting point is labeled A and the goal is labeled B. On a mid-range dual-core laptop, the running time was for each was less than an hour.

To evaluate the challenge of a level, we simulate a player's traversal of the level from the start to the end point, using a standard A* search algorithm. Challenge is determined to be the number of monsters within a given radius at a given point along this traversal path. With this measurement in place, we are able to employ the same fitness function as specified in Section 3.3. Several runs of the algorithm are presented in Figure 4.

With very little extra effort, one can see levels generated for an entirely different type of video game. The generic fitness function has resulted in creating rooms containing clusters of enemies, interspersed with areas containing none, in a manner that could be expected from a human-designed level. A bottom-up, rules-based approach would have necessitated an entirely new set of production rules, but our approach has allowed us to re-use the same genetic algorithm, fitness function, and even some constraints.

5 Discussion

The apparent simplicity of the two examples provided should not obscure the fact that this approach represents a promising alternative to current generative techniques. Firstly, the optimization fitness function allows the intended player experience to be represented explicitly. In other words, instead of specifying how levels are assembled, game designers may simply indicate the particular properties that levels should have. In this way, levels are described declaratively rather than procedurally; instead of treating player experience as an incidental side-effect of the level creation process, the fitness function provides an effective means of handling it directly.

Furthermore, this approach does not exclude the possibility of other, complementary design techniques. Where rule-based generative systems tend to operate in isolation, GAs can work well when used in conjunction with other techniques. As we have shown, game designers are able to hand-craft particular portions of a level without being

required to make any changes to the system. This same idea could be used to interface with other generative systems, either to glue together elements generated elsewhere, or to directly manipulate and optimize the systems themselves.

Another advantage is that this approach is quite modular; constraints, optimizations, and design elements can be added or removed individually. This makes it easier to adjust and test the behaviour of the genetic search.

In a broad sense, this approach factors out the generative algorithm from a particular game artefact. The drive to abstract and generalize implementation details is essential to modern software development, and this practice is used heavily in game development. Features such as 3d animation, physics, and artificial intelligence tend to be handled by third party game engines and are no longer developed from scratch. In the same way that game designers are now able to declaratively specify the specific graphical or physical properties of a given aspect of a game, expecting these properties to be properly handled by the underlying engine, we argue that our approach models a way in which this could be done for level generation. Simple constraints, fun optimization functions, and design elements can be defined for a particular game with no real concern for exactly how these constraints are to be satisfied. Since it is more than likely that games will have many constraints in common (for example, connectivity is a concern in many different types of games), it is possible that these units can be shared among games.

6 Future Work and Conclusion

There are many promising avenues for future work in the area of automated game level generation. Simply continuing to devise constraints and optimization functions for various types of games would likely prove fruitful in evaluating the general applicability of this approach. For example, our current operational definition of fun only accounts for challenge dynamics, which is certainly only one component of fun. A more comprehensive model of fun would need to be employed to account for the presence of game elements not relating directly to challenge, such as collectible rewards. Also, though this paper focuses on proof-of-concept examples rather than on efficiency, a more rigorous comparison of the performance characteristics of the FI-2Pop GA to other possible evolutionary techniques would certainly be worthwhile.

It is also our hope that our method will serve as a useful environment in which to experiment with theoretical conceptualizations of game design. We believe that the ability to explicitly realize models of enjoyment in games will contribute to furthering knowledge in that field. In the same way that simple computational models can serve to elucidate the dynamics of otherwise complex natural phenomena [20], it is possible that models of fun will serve to illustrate fundamental principals of game design. Preliminary work of this nature is introduced in [7], where a model of challenge-based fun in games is explored in more detail. Our generic level generation framework would certainly contribute to this ongoing research.

Even though this work is presently in an exploratory stage, it already exhibits encouraging results and can be viewed as a prototype for a practical tool to assist level designers. Much work is yet to be done, and we anticipate that our general top-down approach to level generation will offer much to the practice and theory of game design.

References

1. Remo, C.: MIGS: Far Cry 2's Guay on the importance of procedural content. Gamasutra (November 2008), http://www.gamasutra.com/php-bin/news_index.php?story=21165
2. Meier, S.: Civilization. MicroProse (1991)
3. The NetHack DevTeam: Nethack (2009), http://www.nethack.org/
4. Kimbrough, S.O., Lu, M., Wood, D.H., Wu, D.J.: Exploring a two-market genetic algorithm. In: GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 415–422. Morgan Kaufmann Publishers Inc., San Francisco (2002)
5. Miyamoto, S., Yamauchi, H., Tezuka, T.: Super Mario Bros. Nintendo (1987)
6. Miyamoto, S., Nakago, T., Tezuka, T.: The Legend of Zelda. Nintendo (1986)
7. Sorenson, N., Pasquier, P.: The evolution of fun: Towards a challenge-based model of pleasure in video games. In: ICCC-X: First International Conference on Computational Creativity, Lisbon, Portugal, pp. 258–267 (2010)
8. Togelius, J., Schmidhuber, J.: An experiment in automatic game design. In: IEEE Symposium on Computational Intelligence and Games, pp. 111–118 (2008)
9. Yannakakis, G., Hallam, J.: Towards capturing and enhancing entertainment in computer games. Advances in Artificial Intelligence, 432–442 (2006)
10. Smith, G., Treanor, M., Whitehead, J., Mateas, M.: Rhythm-based level generation for 2d platformers. In: FDG 2009: Proceedings of the 4th International Conference on Foundations of Digital Games, pp. 175–182. ACM, New York (2009)
11. Pedersen, C., Togelius, J., Yannakakis, G.: Modeling player experience in Super Mario Bros. In: IEEE Symposium on Computational Intelligence and Games (September 2009)
12. Hemert, J.I.: Comparing classical methods for solving binary constraint satisfaction problems with state of the art evolutionary computation. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 82–91. Springer, Heidelberg (2002)
13. 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)
14. Bushnell, N., Bristow, S., Wozniak, S.: Breakout. Atari (1976)
15. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. PhD thesis, Ann Arbor, MI, USA (1997)
16. Sweetser, P., Wyeth, P.: Gameflow: a model for evaluating player enjoyment in games. Comput. Entertain. 3(3), 3 (2005)
17. Salen, K., Zimmerman, E.: Rules of Play: Game Design Fundamentals, October 2003. The MIT Press, Cambridge (2003)
18. Koster, R.: Theory of Fun for Game Design. Paraglyph Press, Scottsdale (2004)
19. Juul, J.: Fear of failing? the many meanings of difficulty in video games. In: Yao, X., Burke, E., Lozano, J.A., Smith, J., Merelo-Guerv, J.J., Bullinaria, J.A., Rowe, J., Tino, P., Kabn, A., Schwefel, H.P. (eds.) The Video Game Theory Reader, vol. 2, pp. 237–252. Routledge, New York (2009)
20. Humphreys, P.: Mathematical modeling in the social sciences. In: Turner, S.P., Roth, P.A. (eds.) The Blackwell guide to the philosophy of the social sciences, pp. 166–184. Wiley-Blackwell, New Jersey (2003)

Search-Based Procedural Content Generation

Julian Togelius[1], Georgios N. Yannakakis[1],
Kenneth O. Stanley[2], and Cameron Browne[3]

[1] IT University of Copenhagen, Rued Langaards Vej 7, 2300 Copenhagen, Denmark
[2] University of Central Florida, 4000 Central Florida Blvd. Orlando, Florida, 32816
[3] Imperial College London, London SW7 2AZ, UK
julian@togelius.com, yannakakis@itu.dk, kstanley@eecs.ucf.edu,
cameron.browne@btinternet.com

Abstract. Recently, a small number of papers have appeared in which the authors implement stochastic search algorithms, such as evolutionary computation, to generate game content, such as levels, rules and weapons. We propose a taxonomy of such approaches, centring on what sort of content is generated, how the content is represented, and how the quality of the content is evaluated. The relation between search-based and other types of procedural content generation is described, as are some of the main research challenges in this new field. The paper ends with some successful examples of this approach.

1 Introduction

In this paper we aim to define *search-based procedural content generation*, investigate what can and cannot be accomplished by the techniques that go under this name, and outline some of the main research challenges in the field. Some distinctions will be introduced between approaches, and a handful of examples of search-based procedural content generation (SBPCG) will be discussed within and classified according to these distinctions. It is important to note that this paper proposes an initial framework of SBPCG approaches that leaves room for further new approaches to be co-located within this young yet emerging field. To begin, procedural content generation is itself introduced.

Procedural content generation (PCG) refers to the creation of game content automatically, through algorithmic means. In this paper, *game content* means all aspects of a game that affect gameplay but are not non-player character (NPC) behaviour or the game engine itself. This definition includes such aspects as terrain, maps, levels, stories, dialogue, quests, characters, rulesets, camera viewpoint, dynamics and weapons. The definition explicitly excludes the most common application of search and optimisation techniques in academic games research, namely, NPC artificial intelligence.

There are several reasons for game developers to be interested in PCG. The first is memory consumption — procedurally represented content can typically be compressed by keeping it "unexpanded" until needed. A good example is the classic space trading and adventure game *Elite* (Acornsoft 1984), which managed

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 141–150, 2010.

to keep hundreds of star systems in the few tens of kilobytes of memory available on the hardware of the day by representing each planet as just a few numbers. Another reason for using PCG is the prohibitive expense of manually creating game content. Many current generation AAA titles employ software such as *SpeedTree* to create whole areas of vegetation based on just a few parameters, saving precious development resources while allowing large, open game worlds.

A third argument for PCG is that it might allow the emergence of completely new types of games, with game mechanics built around content generation. If new content can be generated with sufficient variety in real time then it may become possible to create truly endless games. Further, if this new content is created according to specific criteria, such as its suitability for the playing style of a particular player (or group/community of players) or based on particular types of player experience (challenge, novelty, etc.), it may become possible to create games with close to infinite replay value.

A fourth argument for PCG is that it augments our limited, human imagination. Off-line algorithms might create new rulesets, levels, narratives, etc., which can then inspire human designers and form the basis of their own creations.

2 Dissecting Procedural Content Generation

While PCG in different forms has been a feature of various games for a long time, there has not been an academic community devoted to its study. This situation is now changing with the recent establishment of a mailing list[1], an IEEE CIS Task Force[2], a workshop[3] and a wiki[4] on the topic. However, there is still no textbook on PCG, or even an overview paper offering a basic taxonomy of approaches. Therefore, this section aims to begin to draw some distinctions. Most of these distinctions are not binary, but rather a continuum wherein any particular example of PCG can be placed closer to one or the other extreme. Note that these distinctions are drawn for the purpose of clarifying the role of search-based PCG; of course other distinctions will be drawn in the future as the field matures.

2.1 Online versus Offline

The first distinction to be made is whether content generation is performed online during the runtime of the game, or offline during game development. An example of the former is when the player enters a door to a building and the game instantly generates the interior of the building, which was not there before; in the latter case an algorithm suggests interior layouts that are then edited and perfected by a human designer before the game is shipped. Intermediate cases are possible, wherein an algorithm running on e.g. an RTS server suggests new maps to a group of players daily based on logs of their recent playing styles.

[1] http://groups.google.com/proceduralcontent
[2] http://game.itu.dk/pcg/
[3] http://pcgames.fdg2010.org/
[4] http://pcg.wikidot.com

2.2 Necessary versus Optional Content

A further distinction relating to the generated content is whether that content is necessary or optional. Necessary content is required by the players to progress in the game — e.g. dungeons that need to be traversed, monsters that need to be slain, crucial game rules, and so on — whereas optional content is that which the player can choose to avoid, such as available weapons or houses that can be entered or ignored. The difference here is that necessary content always needs to be correct; e.g. it is not acceptable to generate an intractable dungeon if such an aberration makes it impossible for the player to progress. On the other hand, one can allow an algorithm that sometimes generates unusable weapons and unreasonable floor layouts if the player can choose to drop the weapon and pick another one or exit a strange building and go somewhere else instead.

2.3 Random Seeds versus Parameter Vectors

Another distinction concerning the generation algorithm itself is to what extent it can be parameterised. All PCG algorithms create "expanded" content of some sort based on a much more compact representation. At one extreme, the algorithm might simply take a seed to its random number generator as input; at another extreme, the algorithm might take as input a multidimensional vector of real-valued parameters that specify the properties of the content it generates.

2.4 Stochastic versus Deterministic Generation

A distinction only partly orthogonal to those outlined so far concerns the amount of randomness in content generation, as the variation in outcome between different runs of an algorithm with identical parameters is a design question. It is possible to conceive of deterministic generation algorithms that always produce the same content given the same parameters, but it is well known that many algorithms do not. (Note that we do not consider the random number generator seed a parameter here, as that would imply that all algorithms are deterministic.)

2.5 Constructive versus Generate-and-Test

A final distinction may be made between algorithms that can be called *constructive* and those that can be described as *generate-and-test*. A constructive algorithm generates the content once, and is done with it; however, it needs to make sure that the content is correct or at least "good enough" as it is being constructed. An example of this approach is using fractals to generate terrains [1].

A generate-and-test algorithm incorporates both a generate and a test mechanism. After a candidate content instance is generated, it is tested according to some criteria (e.g. is there a path between the entrance and exit of the dungeon, or does the tree have proportions within a certain range?). If the test fails, all or some of the candidate content is discarded and regenerated, and this process continues until the content is good enough.

3 Search-Based Procedural Content Generation

Search-based procedural content generation (SBPCG) is a special case of the generate-and-test approach to PCG, with the following qualifications:

- The test function does not simply accept or reject the candidate content, but grades it using one *or a vector of* real numbers. Such a test function is sometimes called a *fitness function* and the grade it assigns to the content its *fitness*.
- Generating new candidate content is contingent upon the fitness assigned to previously evaluated content instances; in this way the aim is to produce new content with higher fitness.

All of the examples below (see section 4) use some form of evolutionary algorithm (EA) as the main search mechanism. In an EA, a population of candidate content instances are held in memory. Each generation, these candidates are evaluated by the fitness function and ranked. The worst candidates are discarded and replaced with copies of the good candidates, except that the copies have been randomly modified (i.e. *mutated*) and/or recombined. However, SBPCG does not need to be married to evolutionary computation (EC); other search mechanisms are viable as well. The same considerations about representation and the search space largely apply regardless of the approach to search.

3.1 Content Representation and Search Space

A central question in EC concerns how to represent whatever is evolved. In other words, an important question is how genotypes (i.e. the data structures that are handled by the EA) are mapped to phenotypes (i.e. the data structure or process that is evaluated by the fitness function). An important distinction among representations is between *direct encodings*, wherein the size of the genotype is linearly proportional to the size of phenotype and each part of the genome maps to a specific part of the phenotype, and *indirect encodings*, wherein the genotype maps nonlinearly to the genotype and the former need not be proportional to the latter ([2,3,4]; see [5] for a review).

The study of representations for EC is a broad field in its own right, where several concepts have originated that bear on SBPCG [6]. The problem representation should have the right dimensionality to allow for precise searching while avoiding the "curse of dimensionality" associated with representation vectors that are too large (or the algorithm should find the right dimensionality for the vector). Another principle is that the representation should have a high *locality*, meaning that a small change to the genotype should on average result in a small change to the phenotype and a small change to the fitness value.

Apart from these concerns, of course it is important that the chosen representation is capable of representing all the interesting solutions; this ideal can be a problem in practice for indirect encodings, for which there might be areas of phenotype space to which no genotypes map.

These considerations are important for SBPCG as the representation and search space must be well-matched to the domain if it is to perform optimally. There is a continuum between SBPCG that works with direct and indirect representation. As a concrete example, a maze (for use e.g. in a "roguelike" dungeon adventure game) might be represented:

1. directly as a grid for which mutation works directly on the content (wall, free space, door, monster) of each cell,
2. more indirectly as a list of the positions, orientations and lengths of walls ([7] provides an example),
3. even more indirectly as a repository of different reusable patterns of walls and free space, and a list of how they are distributed (with various transforms such as rotation and scaling) across the grid,
4. very indirectly as a list of desirable properties (number of rooms, doors, monsters, length of paths and branching factor), or
5. most indirectly as a random number seed.

These representations yield very different search spaces. In the first case, all parts of phenotype space are reachable, as the one-to-one mapping ensures that there is always a genotype for each phenotype. Locality is likely high because each mutation can only affect a single cell (e.g. turn it from wall into free space), which in most cases changes fitness only slightly. However, because the length of the genotype would be the number of cells in the grid, mazes of any interesting size quickly encounter the curse of dimensionality.

At the other end of the spectrum, option number 5 does not suffer from search space dimensionality because it searches a one-dimensional space. However, the reason this representation is unsuitable for SBPCG is that there is no locality; one of the main features of a good random number generator is that there is no correlation between the numbers generated by different seed values. All search performs as badly (or as well) as random search.

Options 2 to 4 might all be suitable representations for searching for good mazes. In options 2 and 3 the genotype length would grow with the desired phenotype (maze) size, but sub-linearly, so that reasonably large mazes could be represented with tractably short genotypes. In option 4 genotype size is independent of phenotype size, and can be made relatively small. On the other hand, the locality of these intermediate representations depends on the care and domain knowledge with which each genotype-to-phenotype mapping is designed; both high- and low-locality mechanisms are conceivable.

3.2 Fitness Functions

Once a candidate content item is generated, it needs to be evaluated by the fitness function and assigned a scalar (or a vector of real numbers) that accurately reflects its suitability for use in the game. Designing the fitness function is ill-posed; the designer first needs to decide what, exactly, should be optimized and then how to formalize it. For example, one might intend to design a SBPCG algorithm that creates fun, immersive, frustrating or exciting game content, and thus a fitness

function that reflects how much the particular piece of content contributes to the player's respective affective states while playing. At the current state of knowledge, any attempt to estimate the contribution to "fun" (or affective states that collectively contribute to player experience) of a piece of content is bound to rely on conflicting assumptions. More research is needed at this time to achieve fruitful formalisations of such subjective issues; see [8] for a review.

Three key classes of fitness functions can be distinguished for the purposes of PCG are *direct*, *simulation-based* and interactive fitness functions.

Direct Fitness Functions. In a direct fitness function, some features are extracted from the generated content, and these features are mapped directly to a fitness value. Hypothetical such features might include the number of paths to the exit in a maze, firing rate of a weapon, spatial concentration of resources on an RTS map, and material balance in randomly selected legal positions for board game rule set. The mapping between features and fitness might be linear or non-linear, but typically does not involve large amounts of computation, and is typically specifically tailored to the particular game and content type. This mapping might also be contingent on a model of the playing style, preferences or affective state of the player, meaning that an element of *personalization* is possible. An important distinction within direct fitness functions is between *theory-driven* and *data-driven* functions. In theory-driven functions, the designer is guided by intuition and/or some qualitative theory of player experience to derive a mapping. On the other hand, data-driven functions are based on collecting data on the effect of various examples of content via e.g. questionnaires or physiological measurements, and then using automated means to tune the mapping from features to fitness.

Simulation-based Fitness Functions. It is not always apparent how to design a meaningful direct fitness function for some game content — in some cases, it seems that the content must be sufficiently experienced and operated to be evaluated. An indirect fitness function is based on an artificial agent playing through some part of the game that involves the content being evaluated. Features are then extracted from the observed gameplay (e.g. did the agent win? How fast? How was the variation in playing styles employed?) and used to calculate the fitness of the content. The artificial agent might be completely hand-coded, or might be based on a learned behavioral model of a human player.

Another key distinction is between *static* and *dynamic* simulation-based fitness functions. In a static fitness function, it is not assumed that the agent changes while playing the game; in a dynamic fitness function the agent changes during the game and the fitness value somehow incorporates this change. For example, the implementation of the agent can be based on a learning algorithm and the fitness be dependent on *learnability*, i.e. how well and/or fast the agent learns to play the content that is being evaluated. Other uses for dynamic fitness functions is to capture e.g. order effects and user fatigue.

Interactive Fitness Functions. Interactive fitness functions score content based on interaction with a player in the game, which means that fitness is

evaluated during the actual gameplay. Data can be collected from the player either *explicitly*, using questionnaires or verbal input data, or *implicitly* by measuring e.g. how often or long a player chooses to interact with a particular piece of content [9], when the player quits the game, or expressions of affect such as intensity of button-presses, shaking the controller, physiological response, gaze fixation, speech quality, facial expressions and postures.

3.3 Situating Search-Based PCG

At this point, let us revisit the distinctions in Section 2 and ask how they relate to SBPCG. As stated above, SBPCG algorithms are generate-and-test algorithms. They might take parameter vectors (in particular, parameters that modify the fitness function) or not. As evolutionary and similar search algorithms rely on stochasticity (e.g. a random seed is required for mutation); for the same reasons, these algorithms should be classified as stochastic rather than deterministic.

As there is no general proof that all EAs ultimately converge, there is no guaranteed completion time for a SBPCG algorithm, and no guarantee that it will produce good enough solutions. For these reasons it would seem that SBPCG would be unsuitable for online content generation, and better suited for offline exploration of new design ideas. However, as we shall see later, it is possible to successfully base complete game mechanics on SBPCG, at least if the content generated is optional rather than necessary.

We can also choose to look at the relation between indirect representation and SBPCG from a different angle. If our SBPCG algorithm includes an indirect mapping from genotype to phenotype, this mapping can be viewed as a PCG algorithm in itself, and an argument can be made for why certain types of PCG algorithms are more suitable than others for use as part of an SBPCG algorithm. It is worth noting that some indirect encodings used in various EC application areas bear strong similarities to PCG algorithms for games; several indirect encodings are based on L-systems, as are algorithms for procedural tree and plant generation [3].

4 Case Studies of Search-Based PCG

In this section, we present five examples of search-based procedural content generation, and categorise those according to the distinctions made previously in the paper.

4.1 Rulesets for Pac-Man-like Games

Togelius and Schmidhuber [10] conducted an experiment in which rulesets (necessary content) were evolved offline for grid-based games in which the player moves an agent around, in a manner similar to a discrete version of Pac-Man. Apart from the agent, the grid was populated by walls and "things" of different colours, which could be interpreted as items, allies or enemies depending on the

rules. Rulesets were represented fairly directly as fixed-length parameter vectors, interpreted as the effects on various things when they collided with each other or the agent, and their behaviour. A relatively wide range of games could be represented using this vocabulary, and genotype generation was deterministic except for the starting position of things. The fitness function was dynamic and simulation-based, and completely hand-crafted: an evolutionary reinforcement learning algorithm was used to learn each ruleset and the ruleset was scored dependent on how well it was learned. Games that were impossible or trivial were given low fitness, whereas those that could be learned after some time scored well.

4.2 Rulesets for Board Games

Browne [11] developed a system for offline design of rules (necessary content) for board games using a form of genetic programming. Game rules were represented relatively directly as expression trees, formulated in a custom-designed game description language. This language allowed representation of a sufficiently wide variety of board games, including many well-known games. The EA used for the creation of new rule sets was non-standard in that suboptimal children with poor performance or badly formed rules were not discarded but were instead retained in the population with a lower priority to maintain a necessary level of genetic diversity. The fitness function was a complex combination of direct measures and static simulation-based measures: for example, standard game-tree search algorithms were used to play the generated game as part of the fitness evaluation to investigate issues such as balance and time to play the game. While hand-coded, the fitness function was based on extensive study of existing board games, and measurements of user preferences for board games that exhibited various features.

4.3 Tracks for a Racing Game

Togelius et al. [12] designed a system for offline/online generation of tracks (necessary or optional content, dependent on game design) for a simple racing game. Tracks were represented directly as fixed-length parameter vectors, interpreted deterministically as b-splines (i.e. sequences of Bezier curves) that defined the course of the track. The fitness function was simulation-based, static, and personalised. Each candidate track was evaluated by letting a neural network-based car controller, which had previously been trained to drive in the style of a particular human player, drive on the track. The fitness of the track was dependent on the driving performance of the car: amount of progress, variation in progress and difference between maximum and average speed.

4.4 Weapons for a Space Shooter Game

Hastings et al. [9] developed a multi-player game built on SBPCG. In the game, players guide a spaceship through various parts of space, engaging in fire-fights

with enemies and collecting weapons (each weapon is optional, but having a good set of weapons is necessary for success). Weapons are represented indirectly as variable-size vectors of real values, which are interpreted as connection topologies and weights for neural networks, which in turn control the particle systems that underlie the weapons. The fitness function is interactive, implicit and distributed. Fitness for each weapon depends on how often the various users logged on to the same server choose to fire each weapon relative to how often the weapons sit unused in users' weapon caches.

4.5 Levels and Mechanics for Super Mario Bros

Pedersen et al. [13] modified an open-source clone of the classic platform game *Super Mario Bros* to allow for personalised level and game mechanics generation. Levels were represented very indirectly as a short parameter vector describing mainly the number, size and placement of gaps in the level whereas the sole mechanic investigated was represented as the percentage of the level played from right to left. This vector was converted to a complete level in a stochastic fashion. The fitness function was direct, data-driven and personalised, using a neural network that converted level parameters and information about the player's playing style to one of six emotional state predictors (fun, challenge, frustration, predictability, anxiety, boredom), which could be chosen as components of a fitness function. These neural networks were trained through collecting both gameplay metrics and data on player preferences using variants of the game on a web page with an associated questionnaire.

5 Outlook

As reviewed in the previous section, a small number of successful experiments are already beginning to show the promise of search-based procedural content generation. By classifying these experiments according to the taxonomies presented in this paper, it can be seen both that (1) though all are examples of SBPCG, they differ from each other in several important dimensions, and (2) there is room for approaches other than those that have already been tried; both the type of content generated and the algorithmic approach to generating it may change in the future.

At the same time, there are several hard and interesting research challenges. These include the appropriate representation of game content and the design of relevant, reliable, and computationally efficient fitness functions. The latter challenge in particular is likely to benefit from collaboration with experts from fields other than computational intelligence, including psychology, game design studies and affective computing. The potential gains from providing good solutions to these challenges, however, are significant: the invention of new game genres built on PCG, streamlining of the game development process, and further understanding of the mechanisms of human entertainment are all possible.

Acknowledgements

Thanks to all the participants in the discussions in the Procedural Content Generation Google Group. The research was supported in part by the Danish Research Agency, Ministry of Science, Technology and Innovation; project name: *AGameComIn*; project number: 274-09-0083.

References

1. Miller, G.S.P.: The definition and rendering of terrain maps. In: Proceedings of SIGGRAPH, vol. 20 (1986)
2. Bentley, P.J., Kumar, S.: The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 35–43 (1999)
3. Hornby, G.S., Pollack, J.B.: The advantages of generative grammatical encodings for physical design. In: Proceedings of IEEE CEC (2001)
4. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines Special Issue on Developmental Systems 8(2), 131–162 (2007)
5. Stanley, K.O., Miikkulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2), 93–130 (2003)
6. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. Springer, Heidelberg (2006)
7. Ashlock, D., Manikas, T., Ashenayi, K.: Evolving a diverse collection of robot path planning problems. In: Proceedings of IEEE CEC, pp. 6728–6735 (2006)
8. Yannakakis, G.N.: How to Model and Augment Player Satisfaction: A Review. In: Proceedings of the 1st Workshop on Child, Computer and Interaction, Chania, Crete. ACM Press, New York (2008)
9. Hastings, E., Guha, R., Stanley, K.O.: Evolving content in the galactic arms race video game. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2009)
10. Togelius, J., Schmidhuber, J.: An Experiment in Automatic Game Design. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, Perth, Australia, pp. 252–259. IEEE, Los Alamitos (2008)
11. Browne, C.: Automatic generation and evaluation of recombination games. PhD thesis, Queensland University of Technology (2008)
12. Togelius, J., De Nardi, R., Lucas, S.M.: Towards automatic personalised content creation in racing games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2007)
13. Pedersen, C., Togelius, J., Yannakakis, G.N.: Modeling Player Experience in Super Mario Bros. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, Milan, Italy, pp. 132–139. IEEE, Los Alamitos (2009)

Evolution of Grim Trigger in Prisoner Dilemma Game with Partial Imitation

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

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

Abstract. The emergence of Grim Trigger as the dominant strategy in the Iterated Prisoner Dilemma (IPD) on a square lattice is investigated for players with finite memory, using three different kinds of imitation rule: the traditional imitation rule where the entire data base of the opponent's moves is copied, and the two more realistic partial imitation rules that copy only a subset of opponent's moves based on information of games played. We find that the dominance of Grim Trigger is enhanced at the expense of some well known strategies such as tit-for-tat (TFT) when a player has access only to those moves observed in past games played with his opponents. The evolution of the clusters of Grim Trigger in the early stage of the games obeys a common pattern for all imitation rules, before these clusters of Grim Triggers coalesce into larger patches in the square lattice. A physical explanation for this pattern evolution is given. Implication of the partial imitation rule for IPD on complex networks is discussed.

1 Introduction

Evolutionary game [1,2,3,4] provides a rich playground for the simulation of multi-agent systems with complex dynamics revealed through the evolving patterns of various strategies used by the players. These spatial-temporal patterns are of interest to many scientists working in various fields, ranging from computer science, physics, ecology and biology. One of the most studied games by political scientists and sociologists is the Prisoner's Dilemma, as it provides a simple model of the difficulties of cooperation [5,6,7] in a world populated by egoists. In the Prisoner Dilemma game (PD) two players can choose to cooperate (C) or defect (D). Each player will gain a payoff depending jointly on his choice and the opponent's choice. Cooperation yields a payoff $R(S)$ if the opponent cooperates (defects) and defection yields $T(P)$ if the opponent cooperates (defects). We call R the *Reward* for cooperation, S the *Sucker*'s payoff, T the *Temptation* to defect and P the *Punishment*. Typically, $T > R > P > S$ and $2R > T + P$. The Prisoner Dilemma game is a non zero sum game because one

* Corresponding Author.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 151–160, 2010.

player's loss does not equal the opponent's gain. For player without memory, the best strategy for a selfish individual is to defect, although this will result in mutual defection and lead to the worst collective effect for the society. In this PD game, the expectation of defection (D) is greater than the expectation of cooperation (C), independent of the opponent's strategy, even though cooperation yields a higher total payoff for the society. In order to further investigate the emergence of cooperation, a variant of the PD game is the spatial PD game (SPDG), which describes the evolution pattern for a set of players fixed on a lattice, with each player playing the PD game with nearest neighbors. Since now there is a spatial restriction on the players, cooperators can support each other [8,9] and enhance the survival of cooperators [6,10]. For the SPDG, the problem can be mapped onto the statistical mechanics of the two-state Potts model Hamiltonian [2,11] that describes the total income of player i by

$$H_i = \sum_{j(i)} \underset{\sim}{S}_i^T \underset{\sim}{A} \underset{\sim}{S}_j \quad \text{with} \underset{\sim}{S}_i^T,\ \underset{\sim}{S}_j \in \left\{ \overrightarrow{C}, \overrightarrow{D} \right\} \text{and} \overrightarrow{C} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ \overrightarrow{D} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad (1)$$

Here $\underset{\sim}{S}_j$ is the state vector for the player j who is the neighbor of player i and the state vector can be either one of the two unit vectors $\left\{ \overrightarrow{C}, \overrightarrow{D} \right\}$. The summation runs over all the neighbors of the player i sitting at node i, while the neighborhood is defined by the topology of the given network. In the PD game, complication arises for players with the ability to remember a fixed number of the most recent events and supply each player with a set of answers to respond to every possible given history of the game. We call such an answer a "Move". The finite history of the responses of the players is recorded. The rule that describes what move should be used given a particular history of interaction is called a *Strategy*. A complete strategy should include an answer to every possible situation. Players will adapt their strategies, imitating other more successful players following certain *Imitation Rule*. Although each player at a particular lattice site has a favorite strategy at time t, he may change to a different strategy at a later time as he realizes from his opponents (his neighbors in SPDG) a better choice. Consequently, the lattice at a particular time can be labeled by a colorful collection of strategies defined on the N sites of the lattice, corresponding to the favorite strategy of the N players at that time. The evolution of this pattern of strategies is one of the main topics of analysis in our present work. Memory is the key that leads to the possible switch of strategy of player i to a new strategy after observing the success of his neighbors, who are his opponents. Without memory, player i will not be able to remember the move of his successful neighbor j, thereby imitating the strategy of player j. Now, the key idea of our model comes from the modification of the traditional imitation rule used in the past research on the PD game. The usual imitation rule assumes that the player will copy the complete strategy of his idol, who is a more successful opponent in his encounter. However, if only a subset of the complete strategy of the idol has been used, then it is unrealistic for the player to copy the whole strategy, including the subset that has never been observed. A realistic modification on the imitation rule is to

copy only those subsets of the complete strategy that have been observed. The modification of the traditional imitation rule is necessitated by the fact that all players can only have finite memory. This observation motivates us to consider a new imitation rule called *partial imitation rule*, as it permits the player to imitate at most the subset of the strategy his idol has used. In real life, a player cannot even remember all the observed moves of his idol. We will formulate our model in Section 2 and the imitation rule in Section 3. The results are discussed in Section 5.

2 Memory Encoding

A two-player PD game yields one of the four possible outcomes because each of the two independent players has two possible moves, cooperate (C) or defect (D). To an agent i, the *outcome* of playing a PD game with his opponent, agent j, can be represented by an ordered pair of moves S_iS_j. Here S_i can be either C for *cooperate* or D for *defect*. In any one game between them: $\{S_iS_j\}$ takes on one of these four outcomes $\{CC, CD, DC, DD\}$. For n games, there will be a total of 4^n possible scenarios. A particular pattern of these n games will be one of these 4^n scenarios, and can be described by an ordered sequence of length $2n$ of the form $S_{i1}S_{j1} \ldots S_{in}S_{jn}$. This particular ordered sequence of outcomes for these n games is called a *history* of games between these two players, which consists of n pairs of outcome $\{S_iS_j\}$, with the leftmost pair being the first game played, while the rightmost pair being the outcome of the last gamed played, or the most recent outcome. (We use capital S to denote the value of either C or D in the history. For example, an ordered sequence of move pairs $DDDDDDCC$ represents that the two players cooperate right after the past three mutual defection $\{DD\}, \{DD\}, \{DD\}$.) We use the convention that the outcome $\{S_iS_j\}$, corresponds to S_i being the move made by agent i, who is the player we address, and S_j is the move made by agent j, the opponent of our player. Depending on the player we address, the representation of the same history is not unique. In SPDG, agent j is one of the neighbors of agent i. We say that a player has a memory of fixed-length m, when this player can remember exactly the outcomes of the most recent m games. A "*Memory*" is a sub-sequence of a history. For example, for an agent i with two-game memory ($m = 2$), will only has a "*Memory*" $DDCC$ given a history represented by $DDDDDDCC$. We encode the memory by a bit string using the convention that cooperation is represented by 1 and defection by 0. Thus, the memory $DDCC$ can be represented by the binary number 0011 or the decimal number 3. The number of all the possible memory, given that the agent can memorize the outcomes of the last m games, is 4^m. (Here 4 refers to the four possible outcomes of one game which is 00, 01, 10, 11). To start the game, let's Consider a non-trivial example when $m = 3$. In this case there are $64 = 4^3$ possible histories of the strategies used by the two players. We need to reserve 1 bit for the first move of our player:{D, C}, and use two more bits for the second move of our player when confronted with the two possibilities of the first move of the opponent

Table 1. Representation of Strategy Sequence in M_1

Memorized History	First Move	DD	DC	CD	CC
Players' Strategy	S_0	S_1	S_2	S_3	S_4

$\{D, C\}$. (Our player can choose C or D when the opponent's first move is D, and our player also can choose C or D when the opponent's first move is C. Thus we need two more bits for our player). To account for the four possible scenarios of the last two moves of the opponents: $\{DD, DC, CD, CC\}$, we need to reserve 4 more bits to record the third move of our player. Thus, for a PD game played by prisoners who can remember 3 games, a player will need $1 + 2 + 4 = 7$ bits to record his first three moves [12]. After this initial stage, the strategy for our player will need to respond to the game history with a finite memory. Since there are a total of $64 = 4^3$ possible Memory, i.e., 64 possible outcomes of the last three games, our player will need 64 more bits. In conclusion, the length of the strategy sequence is $7 + 64 = 71$ and there are a total of $2^{71} - 2.4 \times 10^{21}$ possible strategies. Thus the strategy space for a $m = 3$ game is very large. Let's now denote the ensemble of m-step memory as M_m , then the total number of bits required to encode the possible strategy sequence is $b(m) = 2^m - 1 + 4^m$ and the total number of possible strategies is $|M_m| = 2^{b(m)}$. For $m = 1$, the enumeration of the encoding of the possible strategies shows that there are 32 possible strategies. This can be seen from Table 1 below.

The strategy in M1 can be denoted by $S_0|S_1S_2S_3S_4$. Here the first move is S_0. If the memory is DD, then the move is S_1. If the memory is DC, then the move is S_2. If the memory is CD, then the move is S_3. If the memory is CC, then the move is S_4.

3 Imitation Rule

The standard imitation rule for the spatial PD game without memory is that the focal agent i will adopt the pure strategy of a chosen neighbor depending on payoff. The generalized imitation rule for PD game with memory is adopting the entire set of the complete strategy. We call such imitation rule the traditional imitation rule (tIR). In this way, tIR impose the condition that every agent has complete information about the entire set of the strategy of all its neighbors. Such assumption of complete information is unrealistic since the focal agent only plays a few games with its neighbors while the space of strategies used by the neighbor is generally astronomically larger than F(the number of games played by two agents). A more realistic situation is that the focal agent i only has partial information about the strategy of his neighbors. In this paper, every agent only knows a subset of the strategy used by a chosen neighbor. For a pair of players (i, j), playing F games, the focal player i will only observed a set $(S_j(i, j))$ of moves actually used by agent j. This set $S_j(i, j)$ is usually much smaller than the entire set of possible moves corresponding to the strategy of agent j.

With this partial knowledge of the moves of the neighbors, the new imitation rule for agent i is called the partial imitation rule. We now give an example to illustrate the difference between partial imitation rule and the traditional one for one step memory. Let's consider an agent i with $C|DDDD$ strategy confronts another agent j with the Tit-for-Tat (TFT) strategy ($S_0|S_1S_2S_3S_4 = C|DCDC$) and agent i decides to imitate the agent j's strategy. In tIR, we assume that agent i somehow knows all the five bits of TFT though in the confrontation with agent j only four bits at most of TFT have been used. On the other hand, with partial imitation rule (pIR), when a $C|DDDD$ agent confronts a TFT agent, the $C|DDDD$ will know only four bits of TFT($S_0|S_1S_2S_3S_4 = C|DCDC$), i.e., $S_0 = C$, $S_1 = D$, $S_3 = D$, $S_4 = C$, and S_2 is not applicable as we do not run into S_2 situation since it corresponds to the last pair of moves is DC and our agent i always use D except the first move. Thus, when agent i imitates agent j using pIR, agent i will become ($C|DDDC$), which corresponds to a Grim Trigger (GT) instead of TFT ($C|DCDC$). We call this new imitation rule the type 1 partial imitation rule, denoted by pIR1.

In a more relaxed scenario, we can slightly loosen the restriction on the access of our focal agent i to the information of neighbors' strategy. If we denote the set of agent j's moves used during the confrontation between agent i and agent j as $S_j(i,j)$, then we can assume that agent i knows a larger subset of agent j's strategy, described by

$$G_j(i,j) = \bigcup_{k \in \Omega(j)} S_j(k,j) \tag{2}$$

where $\Omega(j)$ denotes the set of nearest neighbors of agent j. Note that this set of moves used by agent j could be substantially larger than $S_j(i,j)$, but still should generally be much smaller than the entire set of strategy of player j. In pIR1, we provide agent i information on agent j defined by the set $S_j(i,j)$. We now introduce a second type of partial imitation rule, denoted by pIR2, if we replace $S_j(i,j)$ by the larger set $G_j(i,j)$.

Let's illustrate pIR2 with an example. Consider an always-cooperating agent i ($C|CCCC$) confronting a GT ($C|DDDC$) agent j, who has four neighbors, including i. Let's assume that the remaining three neighbors of agent j are always-defecting ($D|DDDD$). Let's call these three neighbors a, b, c. In the confrontation between agent i ($C|CCCC$) and agent j (GT), agent j uses only S_0 and S_4 of Grim Trigger. However, in the confrontation between agent j (GT) and three neighbors other than i (agent a, b and c, who are $D|DDDD$), agent j will use S_0, S_1 and S_3 of Grim Trigger. With pIR1, if agent i imitates agent j, the result is still $C|CCCC$ as they will use C for S_0 and S_4 of Grim Trigger based on the set $S_j(i,j)$. Now, with pIR2, if agent i imitates agent j, i changes from $C|CCCC$ to $C|DCDC$, which is TFT. It is still not a Grim Trigger. Finally, if we use tIR, the traditional imitation rule, we will directly replace agent i with Grim Trigger ($C|DDDC$). This example shows that the result of tIR, pIR1 and pIR2 could be very different, depending on the exact situation.

4 Results of Monte Carlo Simulation

In this paper, agents will be placed on a fixed square lattice of size $L \times L$, with periodic boundary condition. Each agent only interacts with its four nearest neighbors. For one *confrontation* we randomly choose an agent i and a neighbor j of i and let them play a number (F) of games with each other. The reason that in one confrontation, agent i and j have to play $F(>1)$ games is that memory effect will not be evident unless there is some repeated encounter between the two players. In order to test the strategies for different F, we introduce a probability parameter p. In a confrontation, the two players may stop playing with a probability p at the end of a game. We further define one generation of the PD game on the square lattice when $L \times L$ confrontations has been performed. In this paper, for simplicity, we will treat the number of generations passed as the measurement of time (t). With this stopping probability p, one effectively control the average number of games played between pair of players, thereby determining average F. According to (1), the payoff of agent i after playing a game with agent j is given by the interaction term $\underset{\sim}{S}_i^T \underset{\sim}{A} \underset{\sim}{S}_j$. After F games between these two agents, we obtain the average payoff $U(i)$ and $U(j)$ of agent i and j over these games in this confrontation. The payoff parameters used are $T = 5.0, R = 3.0, P = 1.0, S = 0.0$. Agent i will then imitate agent j with a probability $P(S_i \to S_j) = (1 + e^{\beta(U(i)-U(j))})^{-1}$. Here, $1/\beta$ represents the thermal noise level. We use $\beta = 100$.

In order to verify the correctness of our program on SPDG, we first test our algorithm using the traditional imitation rule. We initialize the strategies of every agent with each element assigned "cooperation (C)" or "defection (D)" at equal probability. The result of the simulation, which is shown in Fig.1(a), is very similar to the published result of Baek and Kim (Fig.3a in [13]). Here, TFT and GT dominate at long time. These two strategies together with Pavlov and $C|CCDC$ are the only four surviving strategies in the long run. We then

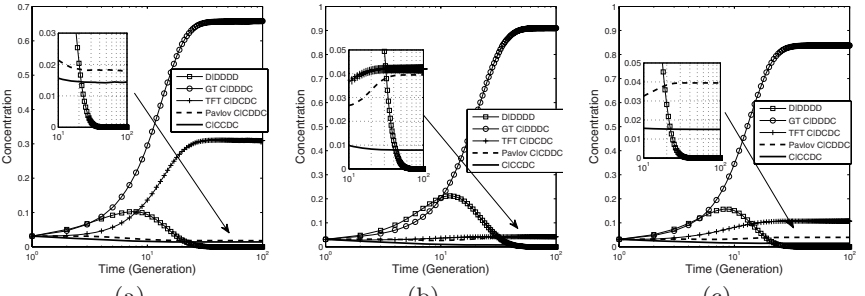


Fig. 1. Concentrations of important strategies in M1 strategy space in SPDG on 100×100 square lattice. Result is averaged over 1000 independent simulations, with $\beta = 100$ using 1(a) traditional Imitation Rule (tIR), 1(b) partial Imitation Rule 1 (pIR1) and 1(c) partial Imitation Rule 2 (pIR2). Practically, 100 runs are enough for producing smooth curves.

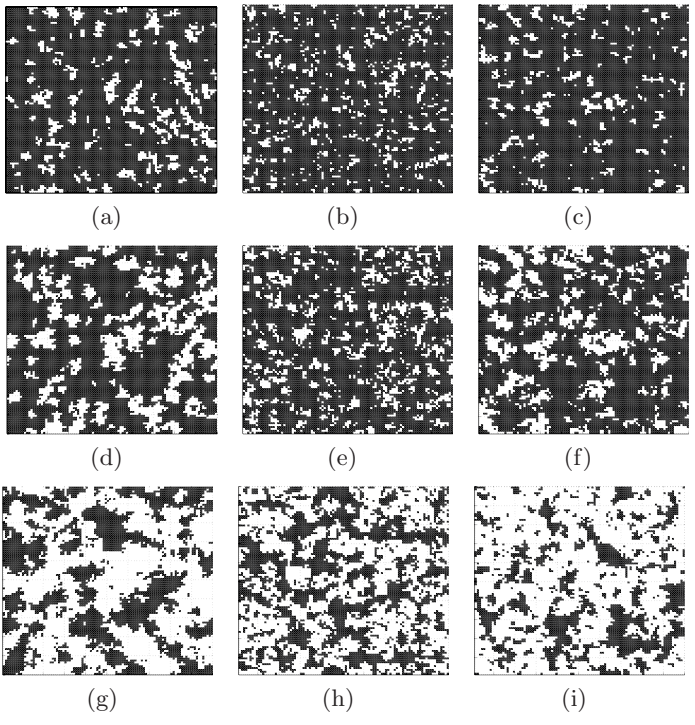


Fig. 2. Evolution patterns of the GT clusters for tIR (2(a),2(d),2(g)), pIR1 (2(b),2(e),2(h)), pIR2 (2(c),2(f),2(i)) at time measured by generation number t = 5, 10, 20. Taken from one typical run for each of the three imitation rules.

use the same program but with the partial imitation rule. In Fig.1(b), we use partial imitation rule 1 (pIR1) and in Fig.1(c), we use pIR2. In both cases, only GT dominates and the concentration of TFT is dramatically reduced to the level of Pavlov and $C|CCDC$. Results are independent of the lattice size. Next, we should note that the average number of games in confrontation is controlled by the probability p. Our numerical experiments show that p affects the concentrations of all the strategies regardless of the imitation rule used. When $p = 1$, agents will always cooperate or defect without making use of the memory mechanism as the game ends with certainty. When p is smaller than 1, there is a finite probability $(1-p)$ that the agents continue playing games, thereby making use of their memory to activate the various moves of their strategies. In general, we should choose p sufficiently small so that the number of games played is sufficiently large and memory effect is evident. As our main concern is on the effect of using partial imitation rule on the dominance of various strategies, we use $p = 0.05$ so that there are about 20 games played in every confrontation. Indeed, we have verified that the general results of our analysis are not sensitive to the values of p, provided that it is smaller than 0.2.

In Fig.2, we show a particular run of the Monte Carlo simulation starting with a randomized initial configuration of players, using three kinds of imitation rules: tIR, pIR1, and pIR2. The time that we make the snapshot are $t = 5$, 10, 20. The white clusters are the players adopting the GT strategy. These clusters grow till they begin to merge into larger clusters. In order to understand the evolution of strategies and the emergence of the dominant clusters of GT, we introduce the following measures for the characterization of the topology of the GT clusters. At a given time, the total number of players adopting the GT strategies can be measured by the total area of the square lattice occupied by GT. Let this total area be $A(t)$. We can also count the length of the boundary between GT and non GT players, and let's denote this boundary as $L(t)$. If we have a single cluster of GT, we can approximate the relation between $L(t)$ and $A(t)$ using a disk of radius $R(t)$, so that $A(t) = \pi(R(t))^2$, $L(t) = 2\pi R(t)$. Now, if there are n equal size disks of GT clusters of radius $R(t)$, then we have $A_n(t) = n\pi(R(t))^2$ and boundary length $L_n(t) = 2n\pi R(t)$. Therefore the number of GT clusters can be estimated to be $n(t) = (L_n(t))^2/(4\pi A_n(t))$. Since both the total area $A_n(t) = n\pi(R(t))^2$ and boundary length $L_n(t) = 2n\pi R(t)$ are measurable, we can obtain the approximate number of GT clusters. Once we obtain $n(t)$, we can obtained the average area of the GT clusters by dividing the total area of GT by $n(t)$: $a(t) = A(t)/n(t)$. Here the total area of GT clusters is denoted by $A(t) \equiv A_n(t)$.

In Fig.3(a), we summarize the results by plotting the average total area of GT players in the 100×100 square lattice as a function of time. We perform this analysis of the GT clusters based on the evolution patterns of the SPDG simulation results using different imitation rules as shown in Fig.2. In Fig.3(b), we observe an interesting universal curve relating the average area $a(t)$ of a GT cluster and the total area $A(t)$ of GT clusters. We see that for all three imitation rules, the data collapse onto the same curve. The collapse of the data is better at small total area, corresponding to the early stage of evolution shown in Fig.3(a): for tIR, the time is less than 25, for pIR2, the time is less than 30, and for pIR1, the time is less than 60. Since the measurement of time is different for different imitation rules, it is easier to measure time of evolution using the total area occupied by GT players. Therefore, the data collapse for the three imitation rules shown in Fig.3(b) indicates some intrinsic scaling relation of the dynamics of the game. Indeed, for tIR, the saturation of the average area of GT clusters in Fig.3(a) occurs sooner at time around 25, since there is a complete knowledge of the opponent's moves before imitation. This saturation effect comes from the coalescence of the various GT clusters to form larger and irregular shaped GT clusters. This phenomenon is shown clearly in Fig.2 for a particular run of the evolution of the GT patterns. When the imitation rule is partial, the knowledge of the possible moves by the GT player is less, so the time needed for the saturation of the average area in Fig.3(a) will be longer for games with partial imitation rule. The fact that the time for saturation for pIR1 is more than pIR2 is then clear, since there is less information on the moves known to the player using pIR1 than pIR2, so saturation occurs sooner in pIR2

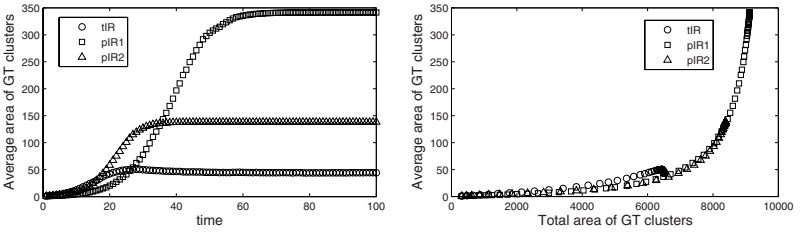


Fig. 3. 3(a) Evolution of the total area occupied by players using GT strategies in the 100×100 square lattice. Time is measured by generation number. 3(b) Average area per GT clusters vs total area of the GT clusters. It shows a collapse of data for three different imitation rules. Averaged from 100 independent simulations.

than in pIR1. When the information on the moves of one's opponent is less available, it will take more time to realize the advantage of the GT strategy, so that the time for saturation of the average area is longer. Thus, in Fig.2, we see that at time $t = 10$, the white clusters for pIR1 (Fig.2(b),2(e),2(h)), which has less information on the opponent's moves, are generally smaller than the white clusters for pIR2 (Fig.2(c),2(f),2(i)), which has more information. For tIR (Fig.2(a),2(d),2(g)), there is complete information, so GT clusters are even larger. After saturation, the system enters into a state of dynamic equilibrium.

5 Conclusion

The memory of the agents has important implication on PD game. In view of the fact that the traditional imitation rule is unrealistic in assuming that a player can copy all the moves of the opponent, we introduce two kinds of partial imitation rules, different by the size of subset of moves observed in past games, and we find very different evolution patterns of various strategies. One major difference is that GT now becomes dominant, and TFT succumbs to the same miserable level of usage as Pavlov. We also observe a universal scaling of the average area of the cluster of GT for all three different imitation rules. This observation implies that there is some hidden scaling relation on the dynamics of SPDG with memory, and the level of partial imitation, as demonstrated by pIR1 and pIR2, corresponds to different region of the universal scaling curve. One generalization that we will further our investigation is to relate the sequence of partial imitation rule to the propagation of information on the moves of an opponent through his interaction with his nearest neighbors, (pIR1 and pIR2), and next nearest neighbors and so on. In this way, a social network based on the propagation of information on the history of moves by this opponent can be established. It will be a very interesting problem to relate this to "rumors propagation" in complex networks. Finally, our analysis indicates that more realistic players in PD game will prefer using GT than TFT, when they use memory and access local information about

the opponent before imitation. This result has important implication of previous studies on PD game as partial knowledge of the opponents' moves should be the norm rather than the exception in real life.

Acknowledgement. K. Y. Szeto acknowledges the support of CERG grant 602506 and 602507.

References

1. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behaviour. Princeton University Press, Princeton (1944)
2. Szabo, G., Fath, G.: Evolutionary games on graphs. Physics Reports 446(4-6), 97–216 (2007)
3. Smith, J.M., Price, G.M.: The logic of animal conflict. Nature 246, 15–18 (1973)
4. Smith, J.M.: Evolution and the Theory of Games. Cambridge University Press, Cambridge (1982)
5. Ohtsuki, H., Hauert, C., Lieberman, E., Nowak, M.A.: A simple rule for the evolution of cooperation on graphs and social networks. Nature 441, 502–505 (2006)
6. Nowak, M.A.: Five Rules for the Evolution of Cooperation. Science 314(5805), 1560–1563 (2006)
7. Axelrod, R.: The Evolution of Cooperation. Basic Books, New York (1984)
8. Nowak, M.A., May, R.M.: The spatial dilemmas of evolution. Int. J. of Bifurcation and Chaos 3(1), 35–78 (1993)
9. Szabo, G., Vukov, J., Szolnoki, A.: Phase diagrams for an evolutionary prisoner's dilemma game on two-dimensional lattices. Phys. Rev. E 72(4), 47107 (2005)
10. Helbing, D., Lozano, S.: Routes to cooperation and herding effects in the prisoner's dilemma game (May 2009)
11. Ariosa, D., Fort, H.: Extended estimator approach for 2x2 games and its mapping to the Ising Hamiltonian. Phys. Rev. E 71, 16132 (2005)
12. Bukhari, S., Adnan, H.A.S.: Using genetic algorithms to develop strategies for the prisoners dilemma. Asian Journal of Information Technology 8(5), 866–871 (2006)
13. 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), 11125 (2008)

Evolving a Ms. PacMan Controller Using Grammatical Evolution

Edgar Galván-López, John Mark Swafford,
Michael O'Neill, and Anthony Brabazon

Natural Computing Research & Applications Group,
University College Dublin, Ireland
{edgar.galvan,john-mark.swafford,m.oneill,anthony.brabazon}@ucd.ie

Abstract. In this paper we propose an evolutionary approach capable of successfully combining rules to play the popular video game, Ms. Pac-Man. In particular we focus our attention on the benefits of using Grammatical Evolution to combine rules in the form of "if <*condition*> then perform <*action*>". We defined a set of high-level functions that we think are necessary to successfully maneuver Ms. Pac-Man through a maze while trying to get the highest possible score. For comparison purposes, we used four Ms. Pac-Man agents, including a hand-coded agent, and tested them against three different ghosts teams. Our approach shows that the evolved controller achieved the highest score among all the other tested controllers, regardless of the ghost team used.

1 Introduction

Ms. Pac-Man, released in early 1980s, became one the most popular video games of all time. This game, the sequel to Pac-Man, consists of guiding Ms. Pac-Man through a maze, eating pills, power pills, and fruit. This task would be simple enough if it was not for the presence of four ghosts that try to catch Ms. Pac-Man. Each ghost has their own, well-defined, behaviour. These behaviors are the largest difference between the Pac-Man and Ms. Pac-Man. In the original Pac-Man, the ghosts are deterministic and players who understand their behavior may always predict where the ghosts will move. In Ms. Pac-Man, the ghosts have non-deterministic elements in their behavior and are not as predictable.

The gameplay mechanics of Ms. Pac-Man are also very easy to understand. When Ms. Pac-Man eats a power pill, the ghosts change their status from inedible to edible (only if they are outside their "nest", located at the centre of the maze) and remain edible for a few seconds. In the edible state they are defensive, and if they are eaten, Ms. Pac-Man's score is increased considerably (the first eaten ghost gives 200 points, the second 400, the third 800, and 1,600 for the last). When all the pills are eaten, Ms. Pac-Man is taken to the next level. Levels get progressively harder by changing the maze, increasing the speed of the ghosts, and decreasing the time to eat edible ghosts. The original version of Ms. Pac-Man presents some very interesting features. For instance, Ms. Pac-Man moves slightly slower than Ghosts when she's eating pills, but she moves slightly faster

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 161–170, 2010.

when crossing tunnels. The most challenging element is the fact that the ghosts' movements are non-deterministic. The goal of the ghosts is to catch Ms. Pac-Man, so they are designed to attack her. Over the last few years, researchers have tried to develop software agents able to successfully clear the levels and simultaneously get the highest score possible (the world record for a human player on the original game stands at 921,360 [4]). The highest score achieved by a computer, developed by Matsumoto [5], based on a screen-capture system that is supposed to be exactly the same as the arcade game, stands at 30,010 [4]. The other top three scores achieved are 15640, 9000 and 8740 points, respectively [5]. It is worth pointing out that all of these methods used a hand-coded approach.

However, it is important to note that there has been work where researchers have used a variety of artificial intelligence approaches to create Ms. Pac-Man players. Some of these approaches state a goal of evolving the best Ms. Pac-Man player possible. Others aim to study different characteristics of an algorithm in the context of this non-deterministic game. Some previous approaches are listed here, but will not be compared against each other due to differences in the Ms. Pac-Man implementation and the goal of the approach.

One of the earliest, and most relevant, approaches comes from Koza [3]. He used genetic programming to combine pre-defined actions and conditional statements to evolve his own, simple Ms. Pac-Man game players. Koza's primary goal was to achieve the highest possible Ms. Pac-Man score using a fitness function that only accounts for the points earned per game. Work similar to [3] is reported by Szita and Lõrincz [10]. Their approach used a combination of reinforcement learning and the cross-entropy method to assist the Ms. Pac-Man agent in "learning" the appropriate decisions for different circumstances in the game. More evolution of Ms. Pac-Man players was carried out by Gallagher [2]. He used a population-based incremental learning approach to help one Ms. Pac-Man player "learn" how to improve its performance by modifying its different parameters. Another, more recent, approach by Lucas [7] uses an evolutionary strategy to train a neural network to play Ms. Pac-Man in hopes of creating the best possible player.

The goal of our work is to successfully evolve rules in the form of "if <*condition*> then perform <*action*>" to maneuver Ms. Pac-Man through the maze, and at the same time, achieve the highest score possible. For this purpose we are going to use Grammatical Evolution (GE) [8,1].

This paper is structured as follows. In the following section we describe how GE works. In Sect. 3 we describe the high-level functions designed to evolve the Ms. Pac-Man agent. In Sect. 4 we describe the experimental setup and Sect. 5 presents the results achieved by our approach, followed by a discussion. Finally, Sect. 6 draws some conclusions.

2 Grammatical Evolution

In GE, rather than representing programs as parse trees, as in Genetic Programming (GP) [3], a variable length linear genome representation is used.

This genome is an integer array with elements called *codons*. A genotype to phenotype mapping process is employed on these integer arrays which uses a user-specified grammar in Backus-Naur Form (BNF) to output the actual phenotype. A grammar can be represented by the tuple $\{N, T, P, S\}$, where N is the set of non-terminals, T is the terminal set, P stands for a set of production rules and, S is the start symbol which is also an element of N. It is important to note that N may be mapped to other elements from N as well as elements from T. The following is an example based on the grammar used in this work (Note: the following is not the actual grammar, just a simplified version; see Fig. 2 for the actual grammar):

```
    Rule                      Productions                    Number
(a) <prog>  ::=  <ifs>  |  <ifs> <elses>                   (0), (1)

(b) <ifs>  ::=  if( <vars> <equals> <vars> ){ <prog> }      (0)
            |   if( <vars> <equals> <vars> ){ <action> }    (1)

(c) <elses>  ::=  else{ <action> }  |  else{ <prog> }      (0), (1)

(d) <action>  ::=  goto(nearestPill)                        (0)
                |   goto(nearestPowerPill)                  (1)
                |   goto(nearestEdibleGhost)                (2)

(e) <equals>  ::=  <  |  <=  |  >                        (0), (1), (2)
                |  >=  |  ==                               (3), (4)

(f) <vars>  ::=  thresholdDistanceGhost                     (0)
              |   inedibleGhostDistance                     (1)
              |   avgDistBetGhosts  |  windowSize          (2), (3)
```

To better understand how the genotype-phenotype mapping process works in GE, here is a brief example. Suppose that we use the grammar defined previously. It is easy to see that each rule has a number of different choices. That is, there are 2, 2, 3, 5, and 4 choices for rules (a), (b), (c), (d), (e), and (f), respectively. Given the following genome: `16 93 34 81 17 46`, we need to define a mapping function (i.e., genotype-phenotype mapping) to produce the phenotype. GE uses the following function: $Rule = \text{c } mod \text{ r}$, where c is the codon integer value and r is the number of choices for the current symbol, to determine which productions are picked for the phenotype. Beginning with the start symbol, `<prog>` , and its definition, `<prog> ::= <ifs> | <ifs> <elses>` the mapping function is performed: $16 \ mod \ 2 = 0$. This means the left-most non-terminal, `<prog>` will be replaced by its 0^{th} production, `<ifs>` , leaving the current phenotype: `<ifs>` .

Because `<ifs>` has two productions and the next codon in the integer array is, 93, `<ifs>` is replaced by: `if( <vars> <equals> <var> ){ <action> }` . Following the same idea, we take the next codon, 34, and left-most non-terminal, `<vars>` and apply the mapping function. The results is 2, so the phenotype is now: `if( avgDistBetGhosts <equals> <var> ) { <action> }`. Repeating the same process for the remaining codons, we have the following expression: `if( avgDistBetGhosts <= inedibleGhostDistance ){goto(nearestPowerPill) }` . It is worth mentioning that in this example, all the codons were used. However, cases may occur where some codons are not used or, during the genotype-phenotype mapping, the end of the genome is reached and there are non-terminals

remaining in the phenotype, causing it to be marked as invalid. If this is the case, there are some options that one can use. For instance, the wrapping operator uses the idea that if a phenotype is incomplete, then the process continues starting from the first codon (from left to right) until a valid phenotype is built or the maximum number of wraps has been reached. If the phenotype is still incomplete at the end of this process, it will be necessary to assign the lowest possible fitness to the individual. As in GP, GE also uses crossover and mutation. The typical form of applying crossover in GE is selecting two genomes and randomly picking a crossover point on each of them. All codons beyond these points are swapped between the genomes. When applying a mutation, it is only necessary to select one genome and then replace a codon at random. It is also possible to direct the search operators like crossover and mutation towards the derivation trees generated during the genotype-phenotype mapping process, and thus operate as per standard GP. In this study genetic operators are applied at the genome level.

3 Our GE Approach to Ms. Pac-Man

As highlighted by the literature there are many approaches one could take when designing a controller for Ms. Pac-Man. We now describe the rule-based approach we've taken. Broadly speaking, a rule is a sentence of the form "if <*condition*> then perform <*action*>". These rules are easy to read, understand, and more importantly, they can be combined to represent complex behaviours.

A number of functions were implemented to be used as primitives in the evolution of the Ms. Pac-Man controller (see Table 1). The aim of each of these functions is to be sufficiently basic, allowing evolution to combine them in a significant manner to produce the best possible behavior for the Ms. Pac-Man controller. In other words, we provide hand-coded, high-level functions and evolve the combination of these functions, pre-defined variables, and conditional statements using GE. These functions were easy to implement, and can be potentially very useful for our purposes. It is worth pointing out that we do not consider these functions to be optimal. For instance, in the case of the `AvoidNearestGhost()` function, we used a *window* that can provide some useful information to Ms. Pac-Man regarding the location of a potential dangerous ghost, but we could have also considered the idea of trying to guess the next position of a ghost given its current location and direction, or keeping track of all available paths in the entire maze given the location of the ghosts. It is also important to mention that these functions are not exclusive. That is, suppose when Ms. Pac-Man has eaten a power pill and is after a ghost, it may take a path full of pills or it can take a path that contains power pills. The latter is not an optimum scenario because it reduces significantly the chances of achieving the highest score possible.

3.1 Hand-Coded Example

The code shown in Fig. 1 calls the functions described in Table 1. It is worth mentioning that we tried different rule combinations with different values for the

Table 1. High-level functions used to control Ms. Pac-Man

Function	*Variable*	*Description*
`NearestPill()`	`npd`	In the original version of this function [4] the agent finds the nearest food pill and heads straight for it regardless of what ghosts are in front of it. We modified it so that in the event a power pill is found before the target food pill, it waits next to the power pill until a different condition is met.
`NearestPowerPill()`	`nppd`	The goal of this function is to go to the nearest power pill.
`EatNearestGhost()`	`ngd`	When there is at least one edible ghost in the maze, Ms. Pac-Man goes towards the nearest edible ghost.
`AvoidNearestGhost()`	`ang`	Calculates the distance of the nearest inedible ghost in a "window" of size $windowSize \times windowSize$, given as a parameter set by evolution, and returns the location of the farthest node from the ghost. This "window" is a mask, where Ms. Pac-Man is at the center.
`NearestInedibleGhost()`	`nig`	Returns the distance from the agent to the nearest inedible ghost. This function is used by the previously explained `AvoidNearestGhost()`.

variables (e.g., windowSize) and the code shown in Fig. 1 gave us the highest score among all the combinations and different values assigned to the variable that we tested. First, we count the number of edible ghosts. Based on this information, Ms. Pac-Man has to decide if it goes to eat power pills, pills, or edible ghosts. We will further explain this hand-coded agent in Sect. 5 where we will compare it with the evolved controller. In the following section, the experimental setup is described to show how GE evolved the combination of the high-level functions described in Table 1.

4 Experimental Setup

We use Lucas' Ms. Pacman simulator [6]. It is important to mention that the simulator only gives one life to Ms. Pac-Man and has only one level. The Ms. Pac-Man implementation was tied into GE in Java (GEVA) [9][1]. This involved creating a grammar that is able to represent what was considered the best possible combination of the high level functions described in Table 1. This grammar can be seen in Fig. 2. The fitness function is defined to reward higher scores. This is done by adding the scores for each pill, power pill, and ghost eaten.

[1] Available from http://ncra.ucd.ie/geva

```
// edibleGhost counts for the number of edible ghosts.
windowSize = 13; avoidGhostDistance = 7; thresholdGhostDistanceGhosts = 10;
inedibleGhostDistance = Utilities.getClosest(current.adj, nig.closest, gs.getMaze());
switch(edibleGhosts){
 case 0:{
   if ( inedibleGhostDistance < windowSize ){
     next = Utilities.getClosest(current.adj, ang.closest, gs.getMaze());
   } else if ( numPowerPills > 0 ) {
     if ( avgDistBetGhosts < thresholdDistanceGhosts ){
       next = Utilities.getClosest(current.adj, nppd.closest, gs.getMaze());
     } else {
         next = Utilities.getClosest(current.adj, npd.closest, gs.getMaze());}

  } else { next = Utilities.getClosest(current.adj, npd.closest, gs.getMaze());}
    break;
  }
  case 1: case 2: case 3: case 4:{
    if ( inedibleGhostDistance < avoidGhostDistance ) {
       next = Utilities.getClosest(current.adj, ang.closest, gs.getMaze());
    }else {
       next = Utilities.getClosest(current.adj, ngd.closest, gs.getMaze()); }
    break;
  }
}
```

Fig. 1. Hand-coded functions to maneuver Ms. Pac-Man

The experiments were conducted using a generational approach, a population size of 100 individuals, the ramped half and half initialisation method, and the maximum derivation tree depth, to control bloat, was set at 10. The rest of the parameters are as follows: tournament selection of size 2, int-flip mutation with probability 0.1, one-point crossover with probability 0.7, and 3 maximum wraps were allowed to "fix" invalid individuals (in case they still are invalid individuals, they were given low fitness values). To obtain meaningful results, we performed 100 independent runs. Runs were stopped when the maximum number of generations was reached.

5 Results and Discussion

5.1 The Best Evolved Controller

The best individual found by GE (Fig. 3) is quite different from the hand-coded agent (Fig. 1). The first thing to notice are the differences in the values of the variables used in the conditional statements. For instance, `windowSize`, which is used by the function `AvoidNearestGhost()` has a different value. When we hand-coded our functions, we set the value at 13, whereas the evolved code set it at 19. Analysing these values, we can see that GE uses a wider window, so Ms. Pac-Man can have more information about the location of ghosts.

Let us continue examining the evolved code. The first condition, `if (edibleGhosts == 0)`, asks if all the ghosts are in an inedible state (in this state Ms. Pac-Man is unable to eat them) if so, it asks if there are power pills available (`numberPowerPills>0`). If this condition holds true, then it executes

```
<prog> ::= <setup><main>
<setup> ::= thresholdDistanceGhosts = <ghostThreshold>; windowSize = <window>;
            avoidGhostDistance = <avoidDistance>; avgDistBetGhosts = (int)adbg.score(gs);
            ang.score(gs, current, windowSize);
<main> ::= if(edibleGhosts == 0){ <statements> } else{ <statements> }
<statements> ::= <ifs> | <ifs> <elses>
<ifs> ::= if( <condition> ) { <action> } | if( <condition> ) { <statements> }
        | if( avgDistBetGhosts <lessX2> thresholdDistanceGhosts ) { <actsOrStats> }
        | if( inedibleGhostDistance <lessX2> windowSize) { <avoidOrPPill> }
<elses> ::= else { <action> } |  else { <statements> }
<actsOrStats> ::= <action> | <statements>
<action> ::= next = getClosest(current.adj, <closest>, gs.getMaze());
           | if ( numPowerPills <more> 0){ <pPillAction> }
             else{ next = getClosest(current.adj, npd.closest, gs.getMaze()); }
<closest> ::= npd.closest | ang.closest | ngd.closest
<avoidOrPPill> ::= <avoidAction> | <pPillAction>
<avoidAction> ::= next = getClosest(current.adj, <avoidClosest>, gs.getMaze());
<pPillAction> ::= next = getClosest(current.adj, <pPillClosest>, gs.getMaze());
<avoidClosest> ::= ang.closest
<pPillClosest> ::= nppd.closest
<condition> ::= <var> <comparison> <var>
<var> ::= thresholdDistanceGhosts | inedibleGhostDistance | avgDistBetGhosts
        | avoidGhostDistance | windowSize
<ghostThreshold> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10
                   | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20
<avoidDistance> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15
<window> ::= 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19
<comparison> ::= <less> | <more> | <lessE> | <moreE> | <equals>
<lessX2> ::= <less> | <lessE>
<less> ::= "<"
<more> ::= ">"
<lessE> ::= "<="
<moreE> ::= ">="
<equals> ::= "=="
```

Fig. 2. The grammar used in our experiments to evolve a Ms. Pac-Man controller using the functions described in Table 1

the `NearestPowerPill()` method. This is quite an interesting sequence of conditions/instructions because it tries to rapidly increase Ms. Pac-Man's score by eating a power pill and then heading to the nearest edible ghost (shown in the second part of conditions/instruction). It is worth noting that this is very different from the previously used, hand-coded approach (see Fig. 1), where it takes a more conservative approach.

If we carefully analyse the last part of the evolved controller (see Fig. 3), we can see that only the last instruction is executed (`next=Utilities.getClosest(current.adj, ngd.closest, gs.getMaze());`). This is because there is a condition that is never met: `if (thresholdDistanceGhosts <= windowSize)`. There is also another element worth mentioning. The function `NearestInedibleGhost()` is never called by the evolved agent. These two elements are absent from the evolved controller indicating that evolution ignored them for the purpose of maneuvering Ms. Pac-Man through the maze. The evolved controller, however, achieved that highest score among all the Ms. Pac-Man agents, as shown in Table 2.

```
thresholdDistanceGhosts = 20; windowSize = 19; avoidGhostDistance = 14; avgDistBetGhosts =
(int) adbg.score(gs, thresholdDistanceGhosts);ang.score(gs, current, windowSize);
if (edibleGhosts == 0) { if (numPowerPills > 0) {
          next = Utilities.getClosest(current.adj, nppd.closest, gs.getMaze()); }
}else { if (thresholdDistanceGhosts <= windowSize) {
          next = Utilities.getClosest(current.adj, ang.closest, gs.getMaze()); }
       else {
          next = Utilities.getClosest(current.adj, ngd.closest, gs.getMaze()); } }
```

Fig. 3. Evolved controller used to guide Ms. Pac-Man

5.2 Benchmarking Performance

In addition to the hand-coded agent and the evolved agent, we used three other Ms. Pac-Man agents (implemented in the code developed by [6]) for comparison purposes. The *Random* agent chooses one of five options (up, down, left, right, and neutral) at every time step. This agent allows reversing at any time. The second agent, called *Random Non-Reverse*, is the same as the random agent except it does not allow Ms. Pac-Man to back-track her steps. Finally, the *Simple Pill Eater* agent heads for the nearest pill, regardless of what is in front of it.

To compare all five different Ms. Pac-Man agents, three ghost teams already implemented in [6] were used. The random ghost team chooses a random direction for each of the four ghosts every time the method is called. This method does not allow the ghosts to reverse. The second team, Legacy, uses four different methods, one per ghost. Three ghosts use the following distance metrics: Manhattan, Euclidean, and a shortest path distance. Each of these distance measures returns the shortest distance to Ms. Pac-Man. The fourth ghost simply makes random moves. Finally, the Pincer team aims to trap Ms. Pac-Man between junctions in the maze paths. Each ghost attempts to pick the closest junction to Ms. Pac-Man within a certain distance in order to trap her.

In Table 2, we show the results for the five different Ms. Pac-Man agents vs. the three different ghost teams, described in the previous paragraph. As expected, the results achieved by these agents versus ghosts are poor. This is not surprising given their nature. It is very difficult to imagine how a controller that does not take into account any valuable information in terms of both, surviving and maximizing the score, can successfully navigate the maze. There are, however, some differences worth mentioning. For instance, random agent shows the poorest performance of all the agents explained previously. This is to be expected mainly because of two reasons: it performs random movements and, more importantly, it allows reversing at any time, so Ms. Pac-Man can easily spend too much time going backwards and forwards in a small space. This is different for the random non-reverse agent that does not allow reversing and as a result of this achieves a higher score. The score achieved by the simple pill eater is better compared with random and random non-reverse agents. This is simply because there is a target of increasing the score by eating pills.

Now, let us take a look at the last two controllers: hand-coded and evolved. The former was designed by the authors in order to achieve the highest score possible. This was done by eating a power pill (if all the ghost are inedible)

Table 2. Results of the five different Ms. Pac-Man agents vs. three different ghost teams over 100 independent runs. Highest scores are shown in boldface.

Ghost Team	*Minimum Score*	*Maximum Score*	*Standard Deviation*	*Sum of all Runs*
	Random Agent			
Random Team	70	810	160.95	24,450
Legacy Team	40	200	31.75	8,670
Pincer Team	40	410	4.33	10,460
	Random Non-Reverse Agent			
Random Team	80	2,800	59.92	89,760
Legacy Team	80	5,310	74.40	69,950
Pincer Team	80	3,810	74.19	73,510
	Simple Pill Eater Agent			
Random Team	240	4,180	108.70	146,010
Legacy Team	250	5,380	107.04	154,720
Pincer Team	240	4,780	96.33	174,370
	Hand-coded Agent			
Random Team	180	11,220	242.68	579,590
Legacy Team	190	11,740	236.58	404,640
Pincer Team	790	12,820	327.10	409,040
	Evolved Agent			
Random Team	480	**11,640**	274.94	428,860
Legacy Team	470	**12,350**	311.60	394,560
Pincer Team	470	**13,830**	405.07	636,180

and then heading straight to the nearest edible ghosts while avoiding inedible ghosts. Once a ghost has been eaten by Ms. Pac-Man, it returns to the ghost nest, resets its status to inedible, and re-enters the maze. The big difference between the hand-coded controller (depicted in Fig. 1) and the evolved controller (shown in Fig. 3) is that the latter takes a more risk-based approach by heading for the power pill (each of these awards 50 points) and then heading for edible ghosts (without taking into account if there are inedible ghosts in the way of Ms. Pac-Man), whereas the former takes a more conservative approach by taking into account the positions of potential dangerous ghosts and if any of these are in the path of Ms. Pac-Man, it tries to avoid the ghost(s). As can be seen in Table 2 the highest score, regardless of the ghost team used, was achieved by the evolved controller.

6 Conclusions

This work proposes a method to evolve high-level functions, described in Table 1, to maneuver Ms. Pac-Man through a maze where the goal is to achieve the highest possible score while avoiding dangerous ghosts. To achieve this goal we used GE for its flexibility in specifying rules in the form of "if <*condition*>

then perform <*action*>". These rules were combined by means of evolution and the resulting evolved controller (Fig. 3) achieved the highest score (Table 2) compared against four other controllers, including a hand-coded controller. All competitors were played against three different ghost teams also described above. As can be seen, the evolved controller is different from the hand-coded controller (shown in Fig. 1) in the sense that the former takes a more risk-based approach whereas the latter is more conservative by checking the positions of ghosts. It is also important to note that the evolved controllers here did not match or exceed the score of Matsumoto's [5] (he used a hand-coded agent). However, this is not discouraging due to the fact that our controller was only allowed one level and one life where Matsumoto's was given three initial lives, could earn more lives, and had more than one level to play.

Acknowledgments

This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

1. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. Springer, Heidelberg (2009), http://www.springer.com/engineering/book/978-3-642-00313-4
2. Gallagher, M.: Learning to play pac-man: An evolutionary, rule-based approach. In: CEC 2003, The 2003 Congress on Evolutionary Computation, pp. 2462–2469. IEEE, Los Alamitos (2003)
3. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
4. Lucas, S.: Ms Pac-Man Competition (September 2009), http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html
5. Lucas, S.: Ms Pac-Man Competition - IEEE CIG 2009 (September 2009), http://cswww.essex.ac.uk/staff/sml/pacman/CIG2009Results.html
6. Lucas, S.: Ms Pac-Man versus Ghost-Team Competition. (September 2009), http://csee.essex.ac.uk/staff/sml/pacman/kit/AgentVersusGhosts.html
7. Lucas, S.: Evolving a neural network location evaluator to play ms. pac-man. In: IEEE Symposium on Computational Intelligence and Games, pp. 203–210 (2005)
8. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Kluwer Academic Publishers, Dordrecht (2003), http://www.wkap.nl/prod/b/1-4020-7444-1
9. O'Neill, M., Hemberg, E., Gilligan, C., Bartley, E., McDermott, J., Brabazon, A.: GEVA - grammatical evolution in java (v 1.0). Tech. rep., UCD School of Computer Science (2008)
10. Szita, I., Lõrincz, A.: Learning to play using low-complexity rule-based policies: illustrations through ms. pac-man. J. Artif. Int. Res. 30(1), 659–684 (2007)

Evolving Bot AI in Unreal™*

Antonio Miguel Mora[1], Ramón Montoya[2], Juan Julián Merelo[1], Pablo García Sánchez[1], Pedro Ángel Castillo[1], Juan Luís Jiménez Laredo[1], Ana Isabel Martínez[3], and Anna Espacia[3]

[1] Departamento de Arquitectura y Tecnología de Computadores, Universidad de Granada, Spain
{amorag,jmerelo,pgarcia,pedro,juanlu,}@geneura.ugr.es
[2] Consejeria de Justicia y Administración Pública, Junta de Andalucíam, Spain
rangel.montoya@juntadeandalucia.es
[3] Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Spain
{amartinez,aesparcia}@iti.upv.es

Abstract. This paper describes the design, implementation and results of an evolutionary bot inside the PC game Unreal™, that is, an autonomous enemy which tries to beat the human player and/or some other bots. The default artificial intelligence (AI) of this bot has been improved using two different evolutionary methods: genetic algorithms (GAs) and genetic programming (GP). The first one has been applied for tuning the parameters of the hard-coded values inside the bot AI code. The second method has been used to change the default set of rules (or states) that defines its behaviour. Both techniques yield very good results, evolving bots which are capable to beat the default ones. The best results are yielded for the GA approach, since it just does a refinement following the default behaviour rules, while the GP method has to redefine the whole set of rules, so it is harder to get good results.

1 Introduction and Problem to Solve

Unreal™ [9] is a First Person Shooter (FPS), an action game in which the player can only see the hands, and the current weapon of his character, and has to fight against enemies by shooting to them. It was developed by Epic Games and launched for PCs in 1998, with great success since it incorporates one of the best multiplayer modes to date. In that mode, up to eight players (in the same PC or connected through a network) fight among themselves, trying to defeat as much of the others (enemies) as possible, getting the so-called *frag* for each defeat, moving in a limited scenario (or map) where some weapons and useful items appear frequently. The players can be human or automatic and autonomous ones, known as *bots*. Each player has a life level which is decreased every time he receives a weapon impact, this decrement depends on the weapon power, the distance, and damaged zone in the character. In addition there are some items that can be used to increase this level or to protect the player.

* Supported in part by the MICYT projects NoHNES (TIN2007-68083) and TIN2008-06491-C04-01, and the Junta de Andalucía (P06-TIC-02025 and P07-TIC-03044).

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 171–180, 2010.

There are many games which include multiplayer modes against human or bots, but there are some features, which made Unreal™ the best framework for us, such as the fact that its *native* bots already have a high level AI, and it offers an own programming language, *UnrealScript*, which is easy, useful and quite powerful. This does not mean that UnrealScript is without drawbacks, including the fact that arrays must be one-dimensional and the limitation in the number of iterations in a loop. These will have to be taken into account when programming a native bot, but in spite of these lacks, it is still the best environment we have found to develop our work.

We should also emphasize that the Unreal™ bot's AI was (and still is) considered as one of the best in all the FPS bots ever designed. It is mainly based on the definition and handling of *states*, each of them modelling the behaviour of the bot when it has a specific status, location in the map, or relationship with the other players (enemies). Many possible states, and many different flow lines between them are defined in a finite state machine [1].

The bot, during a game, changes its current state depending on some factors present in its surroundings and depending on its own status and situation. So, the state change depends most of the times on some specific parameters. They are usually compared with some values, which are most of times hard-coded. This way, the state change (and the power of the Bot's AI) strictly depends on some constant values.

Therefore, the first issue addressed in this work has been the improvement of these constants using a Genetic Algorithm [3], since it is possible to define an array including those parameters and change (evolve) their values to get a better behaviour in the game.

In addition, the finite state machine which determines the bot's AI flow, could be improved too, by avoiding superfluous states or changing the flow between them, in order to arise new transitions between different states. With this objective, a Genetic Programming Algorithm [4] has been applied, since it is an excellent metaheuristic to improve graphs or trees, as is this case.

In both cases we have implemented bots with a *Genetic AI*, or **Genetic Bots (G-Bots)**. We have used evolutionary algorithms to improve the Unreal™ AI given their well-known capacity for optimization.

These way, each G-Bot improves its AI by playing a game; getting a better global behaviour in time, that is, defeating as much enemies as possible (getting frags) and being defeated as less as possible.

2 State of the Art

At the very beginning, the FPS games just included a single player mode (e.g. Wolfenstein in 1987), after this, most of them offered multiplayer possibilities but always against other human players, such as Doom in 1988. The first known game in including autonomous bots (with a simple AI) was Quake in 1992. It presented not only the option of playing against machine-controlled bots, but also the possibility to modify them (just in appearance or in other few aspects) or

create new ones. In order to do this, the programmers could use a programming language called QuakeC, widely used in those years, but which presented some troubles, since it was strictly limited and hard-constrained. So, the bots created using it showed a simple AI (based in fuzzy logic in the best case), and it was not possible to implement more complex techniques as evolutionary algorithms. Unreal™ appeared some years later, being the first game that included an easily programming environment and a more powerful language, so plenty of bots where created. But just a few applied metaheuristics or complex AI techniques, and most of them are based in predefined hard-coded scripts.

Nowadays, there are many games that offer similar possibilities, but almost all of them are devoted to the creation of new maps (sceneries) or characters, being mainly related to the graphical aspect or modifications in their appearance.

In the last few years there have been some studies related to the application of metaheuristics to the behaviour improvement of bots in computer games, as [2,8] where the authors apply Artificial Neural Networks, [6] where evolutionary techniques are used, or [7] in which an evolutionary rule-based system (every individual is a set of conditions, which depend on the bot status and a specific action) has been applied, and which has also been developed under the Unreal™ framework (Unreal Tournament™ 2004).

In the present work we have chosen the original Unreal™ instead of the newer game, since it has a simpler environment but which is enough to perform the proposed study.

3 Genetic Bots

As previously stated in Sect. 1, the objective is to improve the behaviour of an Unreal standard bot, by changing its AI algorithm. Specifically there are two approaches: the first one consist in modifying the default AI by improving the values considered in the conditions assessed to change the current state (and go to another one); the second one is related to the improvement of the finite state machine which the bot's AI follows.

This way, the **Genetic Algorithm-based Bot** (GA-Bot from now on), tries to optimise the values of a set of parameters which represent each one of the hard-coded constants that are in the bot's AI code. These parameters determine the final behaviour of the bot, since most of them are thresholds depending on which, the bot state changes (for instance the distance to an enemy or the bot's health level).

So firstly, it was necessary to determine the parameters to optimise. This way and after a deep analysis of the bot's AI code, 82 parameters were identified. These were too many parameters, since UnrealScript considered as the maximum length for an array 60 floats. In addition it is difficult to evolve such a big array inside a game, since the evaluation function depends on the results of the game, and it would need many individuals and generations to do it.

The number must be reduced, so some parameters were redefined as function of others, and some of the less relevant were unconsidered in the GA individual.

At the end, the array includes just 43 parameters. This set corresponded to an individual in the GA. Thus, each chromosome in the population is composed by 43 genes represented by normalised floating point values (it is a real-coded GA). This way, each parameter moves in a different range, depending on its meaning, magnitude and significance in the game, but all of them are normalised to the [0,1] range. The limits of the range have been estimated, conforming a width interval when the parameters are just modifiers (they are added or subtracted), and a short one if they are considered as the main factor in the bot's decision taking, to avoid an extremely bad behaviour.

This models, in such a way, one approach to the bot's AI. So, the GA evolves 'the behaviour of the bot' (it evolves the trigger values to change between states).

The evaluation of one individual is performed by setting the correspondent values in the chromosome as the parameters for a bot's AI, and placing the bot inside a scenario to fight against other bots (and/or human players). This bot is fighting until a number of global frags is reached, since it is not possible to set a time play for a bot in Unreal, so once the frags threshold has been reached (and the current bot is defeated), the next bot's AI will be the correspondent to the next chromosome in the GA.

Two-point crossover and *simple gene mutation* (change the value of a random gene (or none) by adding or subtracting a random quantity in [0,1]) have been applied as genetic operators (see [5] for a survey). The GA follows the *generational + elitism* scheme, considering as the *selection probability (SP)* for one individual a value calculated using its rank in the population depending on the fitness value, instead of calculating it directly considering the fitness function. This way, it is avoided that superindividuals (those with a very high fitness value) dominate the next population, and a premature convergence occurs. This method is known as *lineal order selection*. Once the SP has been calculated, a probability roulette wheel is used to choose the parents in each cross operation. The elitism has been implemented by replacing a random individual in the next population with the global best at the moment. The worst is not replaced in order to preserve diversity in the population.

The *fitness function* was defined considering the main factors to score a game (through an evaluation function), after some experimentation, they are:

- *frags*, the number of defeated enemies by the bot
- *W*, the number of weapons the bot has picked up
- *P*, the associated power to these weapons
- *I*, the number of items the bot has collected
- *d*, the number of times the bot has been defeated
- *t*, game time the bot has been playing

So, the fitness function equation for the chromosome i is:

$$F_i = \frac{frags_i + [\frac{P_i}{d_i} + (\frac{W_i \cdot 10}{d_i})^{-1}] + \frac{I_i}{10} - \frac{d_i}{10}}{t_i} \quad (1)$$

Where the constant values are used to decrease the relative importance of each term. So, as can be seen, $frags$ is the most important term in the formula.

The following factor (inside square brackets) is related to the weapons; it is composed by two terms: the first one considers the importance of the associated power of the picked up weapons, in average, since a player loses all the weapons once it is defeated. The second term weighs the number of weapons collected by the bot, but it is again an average. In addition this term is inverted since it should take low values when the bot has collected lots of weapons in a life. The objective of this whole factor is to assign a higher weight to a bot which has picked up less but powerful weapons, since searching for them is a risky task, and takes some extra time. The other two factors are devoted to weigh the collected items and the number of times the bot has been defeated (with a negative weight). All the terms are divided by the time the bot has been playing to normalise the fitness of all the chromosomes (they play for a different time).

Since it is the first approach, we have decided to consider an aggregative function instead of a multi-objective one. It is easier to implement and test, and it requires less iterations. The multi-objective approach will be addressed in future works.

The **Genetic Programming-based Bot** (GP-Bot from now on) works in a different way, since it is based in a genetic programming algorithm [4], which can be used to evolve a set of rules. The idea is to redefine the initial set of rules (flows between states) which determines the behaviour of the bot.

The first approach was to work with all the possible inputs and outputs which can be considered, looking at the whole set of states that a default bot can manage, but due to the big amount of them and their complexity, this would mean a huge set of rules to evolve (represented as trees), which can be unapproachable by an algorithm defined inside UnrealScript (due to its strong array size constraints, and limited resources).

So, we decided just to consider the two most important states:

- *Attacking*, in which the bot decides between some possible actions as: search, escape, move in a strategic way, attack and how to do it (from distance, nearby, close, from a flank).
- *Roaming*, in which the bot searches for weapons and items

The flow diagrams of both states are respectively shown in Fig. 1 and Fig. 2.

Then, only the functions applied in the decision taking (since there are some others just used to show an animation in the game, for instance, such as `HitWall`) were studied, in order to get the inputs and the outputs. These functions are devoted to determine the next state to pass, from the one in which the bot currently is, so each one of the functions is divided into a set of sub-functions (*inputs*), and being the *outputs* the correspondent 'jumps' to the next states/substates. For example, there is a function to check if the bot has found an item (`ItemFound`), which returns a 'TRUE' or a 'FALSE' value depending on what the bot has found. It can be considered as an input. One possible output could be `GotoState('Roaming', 'Camp')`, for instance.

This way, all the possible inputs and outputs for these states are used to define rules in the form:

```
IF <INPUT> [AND <INPUT>] THEN <OUTPUT>
```

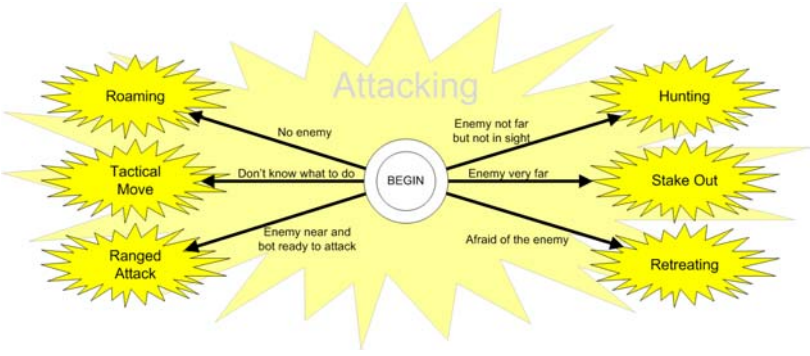


Fig. 1. Flow diagram of Bot's Attacking state. The states are represented by stars.

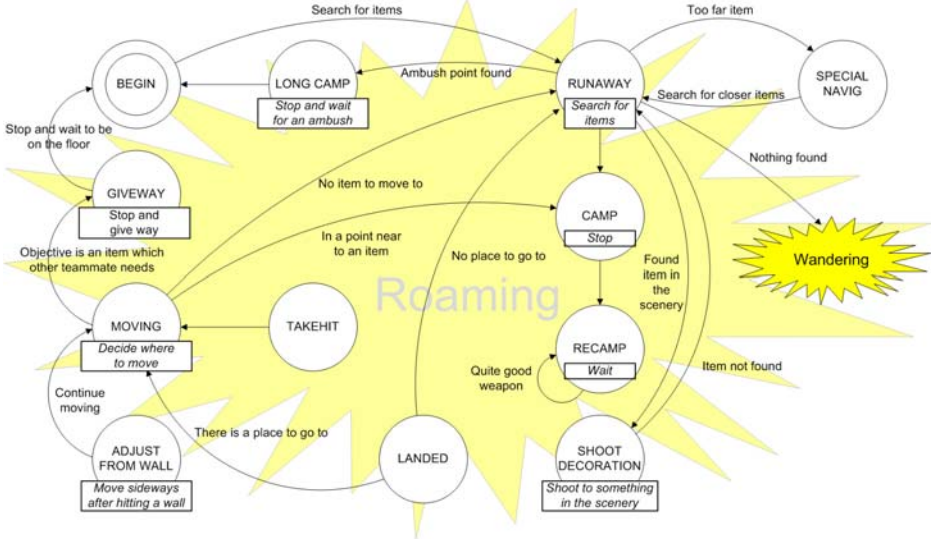


Fig. 2. Flow diagram of Bot's Roaming state. The states are represented by stars, and the sub-states by circles.

Where just two inputs have been considered as a maximum, since after several experimental runs, rules having three (or more) inputs were never triggered.

These rules are modelled as trees, considering as parent nodes the `IF` and `AND` expressions, connected through `RL` nodes (lists); and having as final nodes, the considered input and outputs.

An example tree can be seen in Fig. 3, where a Bot's AI is modelled with four rules:

`IF X1 THEN Y7`, `IF X3 AND X5 THEN Y2`, `IF X8 THEN Y1`, and `IF X9 THEN Y9`

So, every GP-Bot would have an AI structure based in the main set of states, but instead of the two previously mentioned (Attacking and Roaming), they

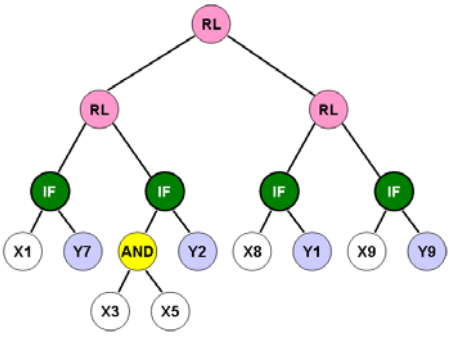


Fig. 3. Example Tree of rules for a Bot's AI. The rules are defined by IF nodes, Xs are inputs and Ys are outputs.

consider one tree of rules, which should be obtained by the evolution of the individuals in the GP algorithm. Thus, every individual in the algorithm is composed by the tree of rules, and also by one chromosome like those considered in the GA-Bot. This way, the evolution is performed over two different AI aspects: the rules, and the parameters, which means it is a *GPGA-Bot*.

During the evolution some genetic operators are used, firstly on the parameters (cross and mutation) which are those presented in the GA-Bot. In addition two specific operators are considered to perform the evolution of the rule trees. The *tree-crossover operator* has been implemented by choosing two different nodes in the parents and interchanging all the sub-tree, below each one of these nodes, taking into account some restrictions to preserve the tree coherence (i.e. not to have an IF node as a final one, or two AND nodes as parent and child). The *tree-mutation operator* just chooses a random node and substitutes the whole sub-tree below it by a randomly generated (but coherent) one. The *selection mechanism* is the same as in the GA-Bot (the lineal order selection).

Relating to the *fitness function*, in this case it should be valued both, the tree of rules and the parameter configuration, so the function has been lightly updated, considering two new factors:

- S, the number of shoots the bot has fired. Included to reward the bots which pick up weapons and use them (some bots do not use them, or do not do it correctly).
- rR, the repetition of rules. Tries to avoid the excessive repetition of rules in the behaviour of a bot, so every 5 repetitions for a rule, it is increased.

So, the fitness function equation for the chromosome i is:

$$F_i = \frac{frags_i + [\frac{P_i}{d_i} + (\frac{W_i \cdot 10}{d_i})^{-1}] + \frac{I_i}{10} - \frac{d_i}{10} + \frac{S_i}{50} - \frac{rR_i}{500}}{t_i} \quad (2)$$

Where all the factors are the same as in Equation 1, excepting the last two, which have been included to evaluate the behaviour in a more accurate way. But neither of these factors has a high relevance.

4 Experiments and Results

We have performed some experiments to test the algorithms. Each of them consists in launching a game match for eight players, being all of them bots[1], and being one of them (its AI) the GA-Bot or the GPGA-Bot.

Each run takes plenty of time (around one hour per generation in average) since every individual in the algorithms (an AI approach), is playing until a number of defeats is reached, and the match is played in real-time. It also depends very much on the map where the bots are fighting so, if it is a big map, it takes longer to reach this number (and change to the next individual). We have considered the parameters showed in Table 1, which have been defined starting from the 'standard' values, and tuning them through systematic experimentation.

Table 1. Parameters of GA and GP-GA algorithms

Number of individuals	30
Mutation Probability	0.01
Crossing Probability	0.6
Number of defeats per chromosome	40

In this work, the experiments have been devoted to test the good behaviour of the algorithms (and also of the bots), since they cannot be compared for the moment with the results yielded by other algorithms.

Four maps have been considered, and five bots of each type (GA and GP-GA) have been tested in each one of these maps. A classical run takes in average around 20 hours for 15 generations, but it depends on the map size.

The algorithms behaviour (related to the fitness) has resulted to be the expected, with some fluctuations in the average due to the classical diversification in the evolutionary algorithms, as can be seen in the Fig. 4 for some examples.

In this figure, a clear evolution in the average fitness is showed for all the cases. This evolution is more marked in the GPGA-Bots, since there are stronger changes in the AI in this algorithm (it evolves the bot's state transition rules), so the improvements in the behaviour are much more obvious, and follow a clear progression. In the GA-Bots this change is less marked since the behaviour is quite similar, but a bit optimised in each generation. These bots show a very coherent behaviour just from the first generation, because their behaviour rules remain the same, and just the decision parameters are changed (evolved). On the other hand, the GPGA-Bots have most of times an incoherent behaviour at the very beginning, since the rules belonging to the main states can be almost random, but the improvements can be more easily noticed after some generations, as has been previously commented.

Relating to the game score, the GA-Bots always beat their rivals, getting a high number of frags in some generations. The GPGA ones cannot get the first position, since the bad behaviour presented in the first generations means low frags, and also being an easy target for their enemies. Two screenshots are

[1] Although it is possible to include also human players.

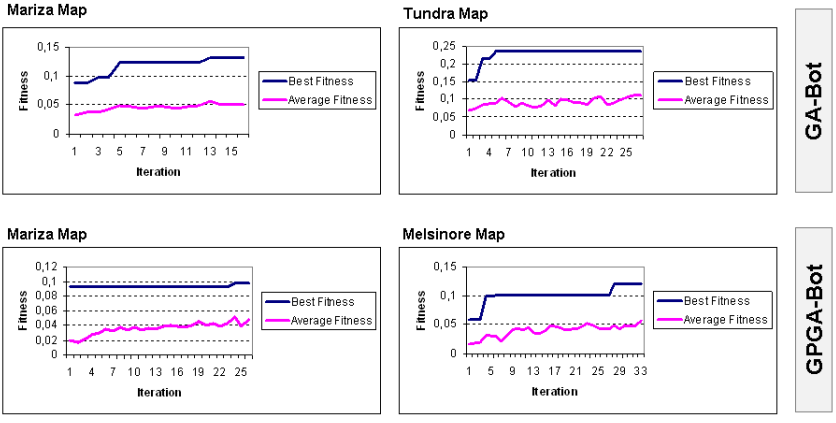


Fig. 4. Example results for two GA-Bots and two GPGA-Bots in two different maps

Fig. 5. Screenshots of the final score (after some generations) for one GA-Bot (GeneticBOT, on the left) and one GPGA-Bot (PGeneticBOT, on the right)

showed in Fig. 5, where it can be seen the classification for both types of bots after some generations.

In both cases we have implemented the final AI configuration (yielded by each one of the algorithms) into two definitive bots, being both of them better than the standard ones presented in Unreal™, winning all the matches.

5 Conclusions and Future Work

In this work, two different evolutionary algorithms have been implemented to improve the AI of the default bots in a PC game named Unreal™. The approaches have been a genetic algorithm, to optimise the decision parameters in the bots; and a genetic programming method, to optimise the set of rules which the bots consider in their AI. Looking at the results, both algorithms work as expected, reaching a clear improvement, and yielding final bot's AI configurations which get the best scores in the matches against the standard bots.

This is our first approach to this problem so there are many future lines of work starting from this point. The first one is the implementation of some

different methods to evolve the AI bots, in order to compare the results with those yielded by the presented algorithms, and also perform some studies to find the best parameter setting for the current algorithms. Another task to address is the implementation of these algorithms inside a newer engine (as Unreal Tournament™), in order to avoid the constraints which obstruct a better problem definition and solving (such as limited arrays and number of iterations in loops).

The third line of improvement is related to the fitness function which is currently an aggregative function, so it could (or should) be separated into different functions, transforming the problem into a multi-objective one, closer to the real problem to address for getting a good bot's AI.

We also want to remark that this is a rather 'noisy' problem, were each of the individuals has a different value for the fitness function at every time, since it depends on many factors which continuously change in time and can be different between two evaluations for the same bot (i.e. the position of the bots while it has been playing, their weapons, the situation of the new weapons, or the position of our bot when it appears in the map), which complicate sometimes the evolution in the algorithms. So maybe a dynamical approach could yield better results.

The last ideas are related to the performance study of a co-evolutionary approach, since it is possible to put in action more than one G-Bot in an scenario. Following the same line, we would also like to implement a cooperative method, where the bots would be grouped into teams which fight between them to get the best results as a whole.

References

1. Booth, T.L.: Sequential Machines and Automata Theory, 1st edn. John Wiley and Sons, Inc., New York (1967)
2. Cho, B.H., Jung, S.H., Seong, Y.R., Oh, H.R.: Exploiting intelligence in fighting action games using neural networks. IEICE - Trans. Inf. Syst. E89-D(3), 1249–1256 (2006)
3. Goldberg, D.E.: Genetic Algorithms in search, optimization and machine learning. Addison-Wesley, Reading (1989)
4. Koza, J.R.: Genetic Programming: On the programming of computers by means of natural selection. MIT Press, Cambridge (1992)
5. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
6. Priesterjahn, S., Kramer, O., Weimer, A., Goebels, A.: Evolution of human-competitive agents in modern computer games. In: IEEE Congress on Computational Intelligence, CEC 2006, pp. 777–784 (2006)
7. Small, R., Bates-Congdon, C.: Agent Smith: Towards an evolutionary rule-based agent for interactive dynamic games. In: IEEE Congress on Evolutionary Computation, CEC 2009, May 2009, pp. 660–666 (2009)
8. Soni, B., Hingston, P.: Bots trained to play like a human are more fun. In: IEEE International Joint Conference on Neural Networks, IJCNN 2008, IEEE World Congress on Computational Intelligence, pp. 363–369 (June 2008)
9. Wikipedia: Unreal — wikipedia, the free encyclopedia (2009), http://en.wikipedia.org/wiki/Unreal

Evolutionary Algorithm for Generation of Entertaining Shinro Logic Puzzles

David Oranchak

http://oranchak.com

Abstract. A Shinro puzzle is a type of deductive reasoning puzzle that originated in Japanese periodicals. To solve the puzzle, one must locate twelve hidden stones on an 8x8 grid using only clues in the form of stone counts per row and column, and arrows placed in the grid that point to some of the hidden stones. Construction of these puzzles by hand is tedious. We explore the use of a simple genetic algorithm that automates construction of Shinro puzzles with desirable qualities which improve their entertainment value.

Keywords: genetic algorithm, logic puzzles.

1 Introduction

Puzzability is a company that specializes in creating and selling a variety of puzzles. In 2007, the company developed and sold a new kind of logic puzzle to Southwest Airlines. The airline began publishing these new puzzles as a regular feature in *Spirit*, their inflight magazine. Named *Shinro*, a Japanese word that means "compass bearing", the puzzle is a simple 8x8 square grid containing twelve holes in unknown locations. Along the top of the grid is an additional row of eight squares. Each square indicates the total number of holes hidden in the column under that square. Similarly, along the left side of the grid is an additional column of eight squares, each indicating the total number of holes hidden in the row to the right of that square. Directional arrows within the puzzle point to some of the hidden holes. Example puzzles are shown in Fig. 1. One solves the puzzle by making a series of reasoned deductions in a process of elimination, identifying squares that *must* contain holes, and squares that *cannot possibly* contain holes. The puzzle solver continues until all twelve holes have been located.

Henceforth, we refer to the holes as "stones" to reflect our own implementation of Shinro.

The popularity of the Shinro puzzles in *Spirit* has led to development of various Shinro video games [1,2,3] and web sites [4,5]. This suggests there is a market for creating new Shinro puzzles. Puzzles are created by selecting hidden stone locations, directional arrow locations, and a desired difficulty in deducing the solution. Construction of these puzzles via automation eliminates the time-consuming and tedious task of designing valid and entertaining puzzles by hand.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 181–190, 2010.

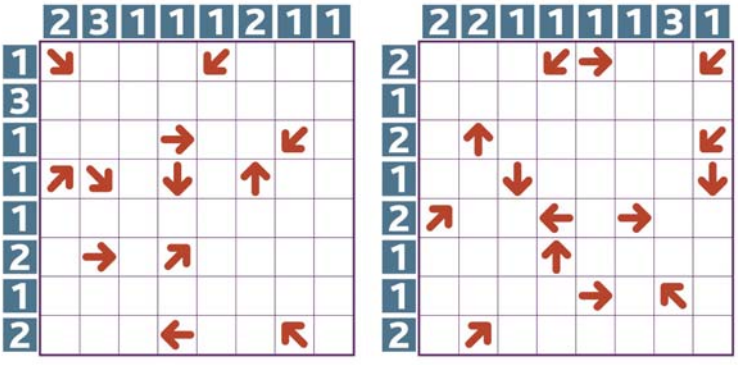


Fig. 1. Sample Shinro puzzles published in *Spirit* magazine [6]

The number of possible Shinro puzzle configurations is astronomical. Each of the 64 positions of an 8x8 puzzle grid can have one of ten possible values (empty, a stone, or one of eight kinds of arrows), bringing the total search space size to 10^{64}. We can reduce this by observing that every valid board must have exactly 12 stones. Thus there are $s =_{64} C_{12} = \frac{64!}{12!(64-12)!}$ possible ways to place stones, and $c = 9^{52}$ remaining possible combinations for the grid positions that lack stones, bringing the total search space down to $s \times c = 8.0 \times 10^{40}$ puzzles. Still much too large for exhaustive searches. A genetic algorithm is suitable for searching such a large space [7]. Mantere and Koljonen were able to show that a GA was an efficient method of generating another kind of constraint satisfaction problem: the sudoku puzzle [8].

2 Methodology

Automatic construction of Shinro puzzles requires development of an algorithm that can automatically solve them. This solver algorithm is used to measure the validity and difficulty of a constructed puzzle, and to estimate its entertainment value. This requires identification of each of the techniques used to solve the puzzles. A similar strategy employed by Ortiz-Garcia et. al. was effective for generating picture-logic puzzles [9].

Stones are located by identifying logical consequences to observations of the puzzle grid state. Similar observations lead to the identification of locations that cannot contain stones. Such locations are marked as "filled". Some of these identifications are easy or trivial to make, while others can be quite challenging. The logical deductions described below are used to automatically solve many Shinro puzzles. The automated solver is the basis for the evolutionary algorithm discussed later.

Count of unfilled positions equals number of remaining stones: If the number of free positions along a row (column) equals the stone count of that

row (column), less the number of stones that have already been placed in that row (column), then these positions must contain stones (Fig. 2).

Row or column count is satisfied: If the number of stones placed in a row (column) exactly matches the stone count for that row (column), then all free positions can be marked as filled (Fig. 3).

Fig. 2. A and B must be locations of stones since the stone count for that row is two

Fig. 3. A, B, and C can be marked as filled because the row count is satisfied by the stone in the third column

Arrow points to only one free square: An *unsatisfied* arrow is an arrow that points to no placed stone. Therefore, there is a hidden stone somewhere along the unsatisfied arrow's path. If there is exactly one free position along the arrow's path, then it must contain a stone (Fig. 4).

One stone remains to be placed, and there is a horizontal or vertical arrow: Consider a row (column) whose stone count, less the number of placed stones along the row (column), is equal to one. If this row (column) contains an unsatisfied horizontal (vertical) arrow, it must point towards the one remaining stone. Therefore, all free positions the arrow points away from cannot contain stones, and thus can be marked as filled (Fig. 5).

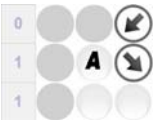


Fig. 4. A has to be a stone. It is the only free position the arrow to its upper right is pointing to.

Fig. 5. A, B, and C cannot possibly be stones, since the horizontal arrow points towards the region of the row that contains the one remaining stone

Locations can be excluded based on identification of non- intersecting arrow paths: This type of deduction identifies regions of rows (columns) in the puzzle in which some number of possible stone placements can be marked as filled due to constraints imposed by unsatisfied arrows pointing into those regions. Figure 6 depicts an example.

Let X denote all free positions within some subset of rows (columns) of the puzzle, where $|X| > 1$. Let n denote the total count of stones remaining to be placed within the rows (columns) of X. Let A denote some subset of unsatisfied

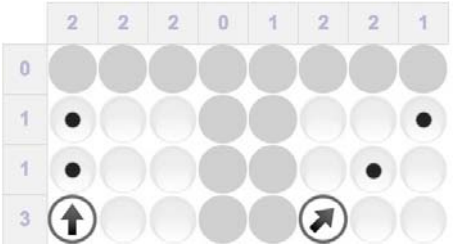


Fig. 6. Two arrows indicate stones along the dotted paths. Since the paths do not intersect, there must be a total of two stones represented by the paths. The covered rows have a total stone count of two. Therefore, none of the non-dotted positions along the covered rows can possibly be a stone, and thus they can all be marked as filled.

arrows of the puzzle. Let us require that each arrow in A has free positions along its path that are contained entirely within X, and that no arrow in A has a path whose free positions intersect the free positions in the path of another arrow in A. Let P denote the set of positions within X that are covered by the free positions along the paths of every arrow in A. If $|A| = n$, then we know that all remaining stones must be located somewhere in positions in P. Therefore, no stones will be found in $X \setminus P$, and these remaining positions can be marked as filled.

These moves can be quite difficult to locate.

Locations can be excluded based on satisfiability of remaining arrows: A free position can be marked as filled if placing a stone there causes an arrow to become unsatisfiable (Fig. 7).

Locations can be excluded based on the *pigeonhole principle*: This is another type of move that can be difficult to locate. In this move, unsatisfied arrows impose constraints on a row (column) such that their satisfaction results in narrowing down the possible choices for filled positions along the row (column). This narrowing of choices for the filled position entails a reduction in the possible locations for remaining stones to be placed in that row (column). Figure 8 depicts an example.

Let X denote a row (column). Let n be the number of stones remaining to be placed in X. Let $m > n$ be the number of unfilled positions in X. Let P be the set of unfilled positions in X, whose column (row) count of remaining stones, less the count of placed stones, is equal to one. A total of $m-n$ positions along X will be marked as filled. We seek $m - n$ unsatisified arrows, A, whose paths contain unfilled positions. Let us require that there is no arrow in A whose unfilled positions intersect the unfilled positions of another arrow in A, or whose unfilled positions intersect P. Let us also impose that every unfilled position represented by A must share the column (row) of a position in P. Thus, satisfaction of an arrow in A identifies a subset of P in which a filled position must be located. Let S be the set of all such subsets. Each time a subset is added to S, the possible

stone positions in X is reduced by one. Once this count reaches $m - n$, then we know that stones must be located in any position in X that is not in P.

Stone placements can be excluded due to impossible scenarios: An attempt to place a stone at a given position can be excluded if the placement results in subsequent moves that lead to an impossible or invalid puzzle state. It is assumed that such brute force attempts to solve Shinro puzzles by exhaustion have little entertainment value. Therefore, we will not consider these moves, except for the simple form shown in Fig. 7.

There may be further types of useful deductions that are not identified above.

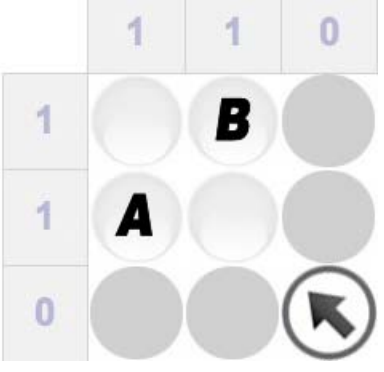


Fig. 7. Placement of a stone at A or B results in an arrow that is impossible to satisfy. Therefore, A and B can be marked as filled.

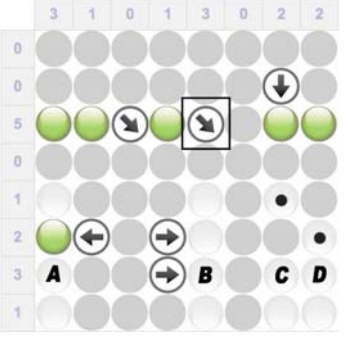


Fig. 8. *Pigeonhole principle*: The indicated arrow points to a stone along the dotted path. Either placement will satisfy its corresponding column count. Therefore, either C or D will be marked as filled. Thus there are only three possible stone placements remaining along the row $ABCD$. Since the filled position must be C or D, and the row count is also three, A and B must be stones.

2.1 Entertainment Value

We assume that completely randomized puzzle configurations are not very fun to play in the long term. We want the generator to create a variety of interesting and entertaining puzzles. Therefore, we need to measure or estimate these qualities. Some of the factors guiding the puzzle designs are listed below.

- We require a degree of control of the difficulty level of generated puzzles. They should neither be too difficult nor too easy to solve. Trivial puzzles and unusually difficult puzzles will quickly tire players.
- Puzzles should have exactly one valid solution.
- We should generate puzzles whose stones and/or arrows form a variety of interesting patterns and symmetries that are more pleasing to view than completely randomized puzzles.

3 Evolutionary Algorithm

A simple genetic algorithm is used to generate the Shinro puzzles. The values of several configuration parameters determine the type of puzzle that is generated. A selection of fitness functions is available to guide the development of an assortment of puzzle varieties. A *target pattern* constraint can be optionally configured, restricting the possible values for some set of positions of the puzzle grid. This permits evolution of puzzles that conform to pre-designed patterns, shapes, and configurations.

3.1 Genome Encoding

Generated puzzles are encoded in an nxn matrix of integer values representing the contents of puzzle grid squares. The possible grid square interpretations are: empty, hidden stone, and arrow pointing in one of eight possible directions.

The optional target pattern constraint can be specified as another nxn matrix of constraint values. This matrix is consulted during population initialization and mutation to prevent insertion of grid square values that violate the given constraints. The constraint types are: Square has no constraint, square must contain an arrow with a specific direction, square must contain a stone, square must be empty, square must contain any arrow, square must contain a stone or any arrow, and square must not be an arrow.

3.2 Initialization

The population of genomes is set to a small size, 10, due to the amount of time needed to perform the fitness evaluation described below. Genomes are initialized to random values. If a target pattern constraint is present, initialization of constrained grid squares is limited to values that do not violate the constraint.

3.3 Genetic Operators

At each generation, a new population is constructed using tournament selection of size three. Elitism is used to prevent loss of good individuals. This is implemented by simply tracking the best evolved individual for the entire run, and copying this individual into each new population.

Crossover is not implemented due to its assumed destructive effect on the satisfaction of constraints of the Shinro puzzles, namely the required fixed number of stones, the set of valid arrows that each point to at least one stone, and the optional target pattern constraint. Further research is necessary to determine the effectiveness of various crossover operators.

When the new population is constructed, mutation is applied by randomly selecting one of a number of available mutators:

- Loop through the puzzle grid and probabilistically change the value of a square. The mutation rate is itself randomly selected from the range $[0, 1]$.

- Swap two randomly selected puzzle grid squares. Repeat n times, where n is randomly selected from the range $[1, 3]$.
- Add an arrow to a randomly selected square.
- Randomly select an arrow and remove it.
- Add a stone to a randomly selected square.
- Randomly select a stone and remove it.

Further, the mutator randomly decides whether or not to enforce symmetry, and whether or not to enforce rotational symmetry. If symmetry is enforced, the selected mutator projects the resulting mutations about the horizontal and vertical axes of the puzzle, both located at the center of the puzzle grid. But if rotational symmetry is enforced, the selected mutator instead projects a puzzle quadrant's mutation into the remaining quadrants by repeatedly rotating the quadrant of the originating mutation by 90 degrees.

The algorithm stops when a minimum number of generations has passed without any improvement to the best genome. The run's statistics and best genome are noted, the population is again reset, and the evolutionary algorithm is repeated. This is done to collect many generated puzzles of high fitness. A final step runs a brute force search on the generated puzzle to ensure that only one unique solution is possible. This computation is not performed during fitness evaluation due to its significant performance impact.

3.4 Fitness Function

The automated solver for generated Shinro puzzles is the basis for fitness computations. The solver inspects each generated puzzle and attempts to solve it using a greedy approach. In the greedy approach, the algorithm looks for easy moves first before proceeding to more difficult moves. Once a move is found, the search for more difficult moves is aborted, and the found move is applied to the puzzle state. The search then repeats until all stones have been located, or no further moves are located. This greedy approach is used to reduce the execution time of the search. The algorithm tracks the counts and types of moves. These counts are used in fitness computations.

We used several fitness functions to evolve a variety a puzzles. Each fitness function shares a common factor: the normalized error count:

$$\epsilon = \frac{1}{1 + |s - s'| + a + v + |m - m'| + e} \tag{1}$$

s is the number of stones required for this puzzle. s' is the number of stones encoded in the genome. a is the number of arrows found that do not point to a stone. If a target pattern constraint is used, v is the number of violated constraints found; otherwise, it is zero. m is the minimum number of solver moves required for acceptance of this generated puzzle. m' is the actual number of moves (of any difficulty level) used by the solver to solve this puzzle. If we are evolving for symmetry, e is the number of grid positions that violate symmetry; otherwise, it is zero.

By itself, this single factor applies evolutionary pressure to locate puzzles that satisfy fundamental constraints. We evolve other desirable puzzle properties by introducing other factors into the fitness function:

- Maximizing the number of moves required to solve the puzzle:

$$f = \epsilon \times [1 - \frac{1}{(1 + steps_d)}] \tag{2}$$

  where $steps_d$ is the number of moves used by the solver to reach a solution. d represents the difficulty factor. When evolving for maximum count of all move types, all move difficulties are counted, and $steps_d$ is equivalent to m'. Otherwise, we only count moves of the specified difficulty level; thus, $steps_d$ may be less than m'.
- Maximizing the clustering of stones and/or arrows:

$$f = \epsilon \times [1 - \frac{1}{(1 + steps_d)}] \times [1 - \frac{1}{1 + c}], \tag{3}$$

$$c = \sum_{i,j} s(i,j), \tag{4}$$

$$s(i,j) = \sum_{u=i-1,v=j-1,(u,v)\neq(i,j)}^{u=i+1,v=j+1} count(u,v) \tag{5}$$

  The following applies only if the item at position (i, j) is the type of item for which we are clustering: $count(u, v)$ returns 1 if the item at (i, j) is found at (u, v) and (u, v) is horizontally or vertically adjacent to (i, j). This represents a *strong* clustering. $count(u, v)$ returns $\frac{1}{4}$ if the item at (i, j) is also found at (u, v) and (u, v) is diagonally adjacent to (i, j). This represents a *weak* clustering. $count(u, v)$ returns 0 if the item at (i, j) is not found at (u, v).

4 Results and Conclusions

The first optimization task was to maximize the number of moves required to solve the generated Shinro puzzles. Figure 9 shows a high-fitness evolved puzzle requiring 37 moves in the automated solver to reach a solution. Despite the high number of moves, the puzzles generated this way remain very easy to solve, because the high numbers of arrows generally give up many clues about stone locations.

Evolution of puzzles with horizontal, vertical, and rotational symmetry of stone and arrow positions was greatly improved by introducing symmetric operations to the standard mutation operators. For instance, instead of changing a single grid value, each of the symmetric grid values are simultaneously changed during a single mutation. Figure 10 shows a generated symmetric puzzle. Figure 11 shows a generated puzzle that has rotational symmetry.

Fig. 9. Evolved puzzle configuration that requires 37 moves to solve

Fig. 10. Puzzle with symmetry. Requires 24 moves to solve.

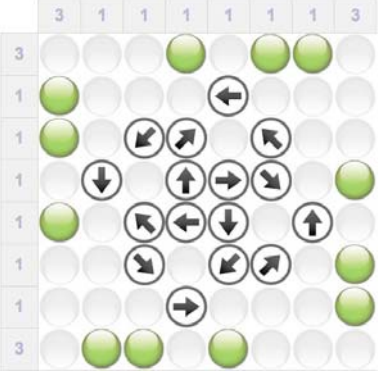


Fig. 11. Puzzle with rotational symmetry, including arrow direction. Requires 30 moves to solve.

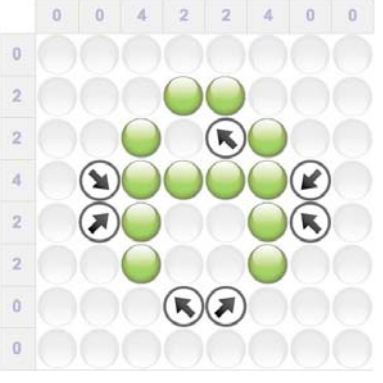


Fig. 12. Puzzle evolved with target pattern constraint: Stones must form "A" shape. Requires 19 moves to solve.

Similarly, when a target pattern constraint is specified, we wanted to prevent mutation from introducing violations to the specified constraint (Fig. 12 depicts a sample constraint that the stones must form the letter "A"). The modified mutation operator checks the constraint prior to changing a grid square value. If the change violates the constraint, the change is suppressed.

To control the difficulty of generated puzzles, several evolutionary runs concentrated on maximizing specific types of moves. The most difficult move represented by the automated solver is the move based on the "pigeonhole" principle. The easiest of any other available move is greedily selected by the automated solver, reflecting the assumed bias of human players to locate the easiest moves first. Pigeonhole moves are not selected unless no other moves are available.

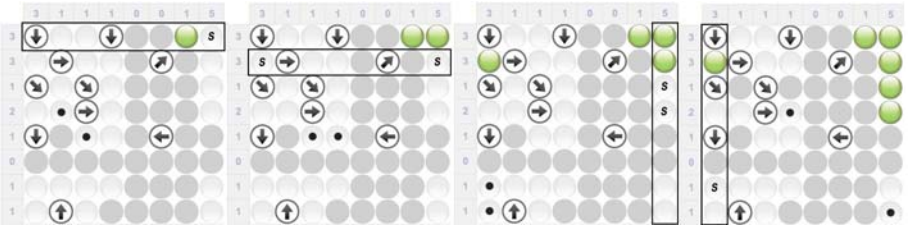


Fig. 13. Evolved puzzle that requires four "pigeonhole" moves. Moves proceed from left to right. Dots indicate sets of possible stone positions that further constrain the indicated row or column. Each *s* indicates the deduced location of a stone due to the necessary placement of another stone somewhere among the dots.

Thus, puzzles that maximize these moves were challenging to discover. One such discovered puzzle, requiring four distinct pigeonhole moves, is depicted in Fig. 13.

The simple genetic algorithm was very effective for the construction of a wide variety of interesting, challenging, and fun Shinro puzzles. Several aspects of our approach warrant further study. The first is the impact of the greedy aspect of the automated solver on evolutionary search. To what extent does the order of moves affect the optimization of puzzle metrics? Another area of future study is the identification of move types that are not represented herein. Can they be discovered by evolutionary search? Would their discovery greatly affect the current evaluation of desirable qualities of generated puzzles? Also, other metrics of desirable puzzle qualities, such as balance of move types, or arrow density, may be useful in further research.

Acknowledgments. I wish to thank Arash Payan, creator of the wonderful iPhone game *Jabeh* which is based on Shinro, and Amy Goldstein of Puzzability, the company that created Shinro puzzles, for their contributions to my research on the background and origins of Shinro puzzles.

References

1. Jabeh - Logic Game for iPhone and iPod Touch, http://jabeh.org/
2. Shinro Mines for iPhone and iPod Touch, http://www.benjiware.com/main/Shinro_Mines.html
3. Shinro - the Next Sudoku (iPhone game), http://web.me.com/romancini/Far_Apps/Shinro.html
4. Shinro Puzzles, http://shinropuzzles.web.officelive.com
5. Sternenhimmel, http://www.janko.at/Raetsel/Sternenhimmel/index.htm
6. Fun and Games. Southwest Airlines Spirit Magazine (2007)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
8. Mantere, T., Koljonen, J.: Solving, rating and generating Sudoku puzzles with GA. In: IEEE Congress on Evolutionary Computation (2007)
9. Ortiz-García, E.G., Salcedo-Sanza, S., Leiva-Murillob, J.M., Pérez-Bellidoa, A.M., Portilla-Figuerasa, J.A.: Automated generation and visualization of picture-logic puzzles. Computers & Graphics 31(5), 750–760 (2007)

Social Learning Algorithms Reaching Nash Equilibrium in Symmetric Cournot Games

Mattheos K. Protopapas[1], Francesco Battaglia[1], and Elias B. Kosmatopoulos[2]

[1] Department of Statistics, University of Rome "La Sapienza",
Aldo Moro Square 5, 00185 Rome Italy
{matteo.protopapas,francesco.battaglia}@uniroma1.it
[2] Department of Production Engineering and Management,
Technical University of Crete, Agiou Titou Square
kosmatop@dssl.tuc.gr

Abstract. The series of studies about the convergence or not of the evolutionary strategies of players that use co-evolutionary genetic algorithms in Cournot games has not addressed the issue of individual players' strategies convergence, but only of the convergence of the aggregate indices (total quantity and price) to the levels that correspond either to the Nash or Walrash Equilibrium. Here we discover that while some algorithms lead to convergence of the aggregates to Nash Equilibrium values, this is not the case for the individual players' strategies (i.e. no NE is reached). Co-evolutionary programming social learning, as well as a social learning algorithm we introduce here, achieve this goal (in a stochastic sense); this is displayed by statistical tests, as well as "NE stages" evaluation, based on ergodic Markov chains.

1 Introduction

The "Cournot Game" models an oligopoly of two or more firms that decide -independently and simultaneously- the quantities they produce and supply to the market. The total quantity produced by all the firms define -via an exogenous demand function- the equilibrium price at the market. The companies' goal is to maximize their profit (1), which is the difference of their sales revenue and their production cost. Co-evolutionary Genetic Algorithms have been used for studying Cournot games, since Arifovic [3] studied the cobweb model. In contrast to the classical genetic algorithms used for optimization, the co-evolutionary versions are distinct at the issue of the objective function. In a classical genetic algorithm the objective function for optimization is given before hand, while in the co-evolutionary case, the objective function changes during the course of play as it is based on the choices of the players.

In the algorithms we use here, each chromosome's fitness is proportional to its profit, as given by

$$\pi(q_i) = Pq_i - c_i(q_i) \tag{1}$$

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 191–200, 2010.

where $c_i(q_i)$ is the player's cost for producing q_i items of product and P is the market price, as determined by all players' quantity choices, from the inverse demand function

$$P = a - b\sum_{i=1}^{n} q_i \tag{2}$$

In Arifovic's algorithms [3], populations are updated after every single Cournot game is played, and converge to the Walrasian (competitive) equilibrium and not the Nash equilibrium [2],[15]. Convergence to the competitive equilibrium means that agents' actions -as determined by the algorithm- tend to maximize (1), with price regarded as given, instead of

$$\max_{q_i} \pi(q_i) = P(q_i)q_i - c_i(q_i) \tag{3}$$

that gives the Nash Equilibrium in pure strategies [2]. Later variants of Arifovic's model [5],[7] share the same properties.

Vriend was the first to present a co-evolutionary genetic algorithm in which the equilibrium price and quantity on the market -but not the strategies of the individual players as we will see later- converge to the respective values of the Nash Equilibrium [16]. In his individual learning, multi-population algorithm, which is one of the two algorithms that we study -and transform- in this article, chromosomes' fitness is calculated only after the chromosomes are used in a game, and the population is updated after a given number of games are played with the chromosomes of the current populations. Each player has its own population of chromosomes, from which he picks at random one chromosome to determine its quantity choice at the current round. The fitness of the chromosome, based on the profit acquired from the current game is then calculated, and after a given number of rounds, the population is updated by the usual genetic algorithm operators (crossover and mutation). Since the populations are updated separately, the algorithm is regarded as individual learning. These settings yield Nash Equilibrium values for the total quantity on the market and, consequently, for the price as well, as proven by Vallee and Yildizoglou [15]. In this study, as well as in [11], we have studied the convergence of the individual agents' choices to the Nash Equilibrium quantities.

Finally Alkemade et al. [1] present the first (single population) social learning algorithm that yields Nash Equilibrium values for the total quantity and the price. The four players pick at random one chromosome from a single population, in order to define their quantity for the current round. Then profits are calculated and the fitness value of the active chromosomes is updated, based on the profit of the player who has chosen them. The population is updated by crossover and mutation, after all chromosomes have been used. As Alkemade et al. [1] point out, the algorithm leads the total quantities and the market price to the values corresponding to the NE for these measures.

2 The Models

In all the above models, researchers assume symmetric cost functions (all players have identical cost functions), which implies that the Cournot games studied are symmetric. Additionally, Vriend [16], Alkemade et al. [1] and Arifovic [3] -in one of the models she investigates- use linear (and decreasing) cost functions. If a symmetric Cournot Game, has in addition, indivisibilities (discrete, but closed strategy sets), it is a pseudo-potential game [6] and the following theorem holds:

Theorem 1. *"Consider a n-player Cournot Game. We assume that the inverse demand function P is strictly decreasing and log-concave; the cost function c_i of each firm is strictly increasing and left-continuous; and each firm's monopoly profit becomes negative for large enough q. The strategy sets S^i, consisting of all possible levels of output producible by firm i, are not required to be convex, but just closed. Under the above assumptions, the Cournot Game has a Nash Equilibrium [in pure strategies]"* [6].

This theorem is relevant when one investigates Cournot Game equilibrium using Genetic Algorithms, because a chromosome can have only a finite number of values and, therefore, it is the discrete version of the Cournot Game that is investigated, in principle. Of course, if one can have a dense enough discretization of the strategy space, so that the NE value of the continuous version of the Cournot Game is included in the chromosomes' accepted values, it is the case for the NE of the continuous and the discrete version under investigation to coincide.

In all three models we investigate in this paper, the assumptions of the above theorem hold, and hence there is a Nash Equilibrium in pure strategies. We investigate those models for the cases of $n = 4$ and $n = 20$ players.

The first model we use is the linear model used in [1]: The inverse demand is given by

$$P = 256 - Q \tag{4}$$

with $Q = \sum_{i=1}^{n} q_i$, and the common cost function of the n players is

$$c(q_i) = 56q_i \tag{5}$$

The Nash Equilibrium quantity choice of each of the 4 players is $\hat{q} = 40$ [1]. In the case of 20 players we have, by solving (3), $\hat{q} = 9.5238$. The second model has a polynomial inverse demand function.

$$P = aQ^3 - b \tag{6}$$

and linear symmetric cost function

$$c = xq_i + y \tag{7}$$

If we assume $a < 0$ and $x > 0$ the demand and cost functions will be decreasing and increasing, respectively, and the assumptions of theorem (1) hold. We set $a = -1$, $b = 7.36 \times 10^7 + 10$, $x = y = 10$, so $\hat{q} = 20$ for $n = 20$ and $\hat{q} = 86.9401$ for $n = 4$.

Finally, in the third model, we use a radical inverse demand function

$$P = aQ^{\frac{3}{2}} + b \tag{8}$$

and the linear cost function (7). For $a = -1$, $b = 8300$, $x = 100$ and $y = 10$ theorem (1) holds and $\hat{q} = 19.3749$ for $n = 20$, while $\hat{q} = 82.2143$ for $n = 4$.

3 The Algorithms

We use two multi-population (each player has its own population of chromosomes representing its alternative choices at any round) co-evolutionary genetic algorithms, Vriend's individual learning algorithm [16] and co-evolutionary programming, a similar algorithm that has been used for the game of prisoner's dilemma [10] and, unsuccessfully, for Cournot Duopoly [13]. Since those two algorithms don't, as it will be seen, lead to convergence to the NE in the models under consideration, we introduce two different versions of the algorithms, as well, which are characterized by the use of opponent choices, when the new generation of each player's chromosome population is created, and therefore can be regarded as "socialized" versions of the two algorithms. The difference between the "individual" and the "social" learning versions of the algorithms is that in the former case the population of each player is updated on itself (i.e. only the chromosomes of the specific player's population are taken into account when the new generation is formed), while on the latter, all chromosomes are copied into a common "pool", then the usual genetic operators (crossover and mutation) are used to form the new generation of that aggregate population and finally each chromosome of the generation is copied back to its corresponding player's population. Thus we have "social learning", since the alternative strategic choices of a given player at a specific generation, as given by the chromosomes that comprise its population, are affected by the chromosomes (the ideas should we say) all other players had at the previous generation.

Co-evolutionary programming [13] is quite similar, with the difference that the random match-ups between the chromosomes of the players' population at a given generation are finished when all chromosomes have participated in a game; and then the population is updated, instead of having a parameter (GArate) that defines the generations at which populations update takes place.

In our implementation, we don't use elitism. The reason is that by using only selection proportional to fitness, single (random) point crossover and finally, mutation with fixed mutation rate for each chromosome bit throughout the simulation, we ensure that the algorithms can be classified as *canonical economic GA's* [13], and that their underlying stochastic process form an ergodic Markov Chain [13].

In order to ensure convergence to Nash Equilibrium, we introduce the two "social" versions of the above algorithms. Vriend's multi-population algorithm could be transformed to:

1. A set of strategies [chromosomes representing quantities] is randomly drawn for each player.

2. While $Period < T$
   (a) (If $Period \bmod GArate = 0$): Use GA procedures (roulette wheel selection, single, random point crossover and mutation), to create a new generation of chromosomes, from a population consisting of the chromosomes belonging to the union of the players' populations. Copy the chromosomes of the new generation to the corresponding player's population, to form a new set of strategies for each player.
   (b) Each player selects one strategy. The realized profit is calculated (and the fitness of the corresponding chromosomes, is defined, based on that profit).

And social co-evolutionary programming is defined as:

1. Initialize the strategy population of each player
2. Choose one strategy of the population of each player randomly from among the strategies that have not already been assigned profits. Input the strategy information to the tournament. The result of the tournament will decide profit values for these chosen strategies.
3. Repeat step (2) until all strategies are assigned a profit value.
4. Apply the evolutionary operators (selection, crossover, mutation) at the union of players' populations. Copy the chromosomes of the new generation to the corresponding player's population to form the new set of strategies.
5. Repeat steps (2)-(4) until maximum number of generations has been reached.

So the difference between the social and individual learning variants is that chromosomes are first copied in an aggregate population, and the new generation of chromosomes is formed from the chromosomes of this aggregate population. From an economic point of view, this means that the players take into account their opponents choices when they update their set of alternative strategies. So we have a social variant of learning, and since each player has its own population, the algorithms should be classified as "social multi-population economic Genetic Algorithms" [12],[13]. It is important to note that the settings of the game allow the players to observe their opponent choices after every game is played, and take them into account, consequently, when they update their strategy sets.

It is not difficult to show that the stochastic process of all the algorithms presented here form a regular Markov chain [9]. Having a Markov chain implies that the usual performance measures -namely mean value and variance- are not adequate to perform statistical inference, since the observed values in the course of the genetic algorithm are inter-dependent. In a regular Markov chain however, one can estimate the limiting probabilities of the chain by estimating the components of the fixed frequency vector the chain converges to, by

$$\hat{\pi_i} = \frac{N_i}{N} \tag{9}$$

where N_i is the number of observations in which the chain is at state i and N is the total number of observations [4]. In the algorithms presented here, assuming n players, with k chromosomes consisting of l bits in each player's population, the total number of possible states is 2^{knl}, making the estimation of the limiting probabilities of all possible states, practically impossible. On the other hand,

one can estimate the limiting probability of one or more given states, without needing to estimate the limiting probabilities of all the other states. A state of importance could be the state where all chromosomes of all populations represent the Nash Equilibrium quantity (which is the same for all players, since we have a symmetric game). We call this state *Nash State*.

4 Simulation Settings

We use two variants of the three models in our simulations. One about $n = 4$ players and one having $n = 20$ players. We use 20-bits chromosomes for the $n = 4$ players case and 8-bits chromosomes for the $n = 20$ case. A usual mechanism [3],[16] is used to transform chromosome values to quantities. After an arbitrary choice for the maximum quantity, the quantity that corresponds to a given chromosome is given by:

$$q = \frac{1}{q_{max}} \sum_{k=1}^{L} q_{ijk} 2^{k-1} \tag{10}$$

where L is the length of the chromosome and q_{ijk} is the value of the k_{th} bit of the given chromosome (0 or 1). According to (10) the feasible quantities belong in the interval $[0, q_{max}]$. By setting

$$q_{max} = 3\hat{q} \tag{11}$$

where $\hat{q}$ is the Nash Equilibrium quantity of the corresponding model, we ensure that the Nash Equilibrium of the continuous model is one of the feasible solutions of the discrete model, analyzed by the genetic algorithms, and that the NE of the discrete model will be therefore, the same as the one for the continuous case. And, as it can be easily proven by mathematical induction, that the chromosome corresponding to the Nash Equilibrium quantity, will always be $0101 \ldots 01$, provided that chromosome length is an even number.

The *GArate* parameter needed in the original and the "socialized" versions of Vriend's algorithms, is set to $GArate = 50$, an efficient value suggested in the literature [15],[16]. We use single - point crossover, with the point at which chromosomes are combined [8] chosen at random. Probability of crossover is always set up to 1, i.e. all the chromosomes of a new generation are products of the crossover operation, between selected parents. The probability of mutating any single bit of a chromosome is fixed throughout any given simulation -something that ensures the homogeneity of the underlying Markov process. The values that have been used (for both cases of $n = 4$ and $n = 20$) are

$$p_m = 0.1, 0.075, \ldots, 0.000025, 0.00001.$$

We used populations consisting of

$$pop = 20, 30, 40, 50$$

chromosomes. These choices were made after preliminary tests that evaluated the convergence properties of the algorithms for various population choices, and

they are in accordance to the population sizes used in the literature ([16],[1], etc.).

Finally, the maximum number of generations that a given simulation runs, were

$$T = 10^3, 2 * 10^3, 5 * 10^3, 10^4, 2 * 10^4, 5 * 10^4$$

Note that the number of total iterations (number of games played) of Vriend's individual and social algorithms is $GArate$ times the number of generations, while in the co-evolutionary programming algorithms is number of generations times the number of chromosomes in a population, which is the number of match-ups.

We run 300 independent simulations for each set of settings for all the algorithms, so that the test statistics and the expected time to reach the Nash Equilibrium (NE state, or first game with NE played), are estimated effectively.

5 Synopsis of Results

Although the individual - learning versions of the two algorithms led the estimated expected value of the average quantity (as given in eq.(12))

$$\bar{Q} = \frac{1}{nT} \sum_{t=1}^{T} \sum_{i=1}^{n} q_{it} \tag{12}$$

(T = number of iterations, n = number of players), close to the corresponding average quantity of the NE, the strategies of each one of the players converged to different quantities. The trajectory of the average market quantity in Vriend's algorithm

$$Q = \frac{1}{n} \sum_{i=1}^{n} q_{it} \tag{13}$$

(calculated in (13) is quite similar to the trajectory of the same measure in the co-evolutionary case. The estimated average values of the two measures (eq.(12)) were 86.2807 and 88.5472 respectively, while the NE quantity in the polynomial model (6) is 86.9401. The unbiased estimators for the standard deviations of the Q (eq.(14)) were 3.9776 and 2.6838, respectively.

$$s_Q = \frac{1}{T-1} \sum_{i=1}^{T} (Q_i - \bar{Q})^2 \tag{14}$$

The estimators of the mean values of each player's quantities, as calculated by eq.(15),

$$\bar{q}_i = \frac{1}{T} \sum_{i=1}^{T} q_i \tag{15}$$

are given on table 1.

That significant difference between the mean values of players' quantities was observed in all simulations of the individual - learning algorithms, in all models

Table 1. Mean Players' Quantities for $n = 4$ players, $pop = 50$, $GArate = 50$, $p_{cr} = 1$, $p_{mut} = 0.01$, $T = 2,000$ generations

Player	Vriend's algorithm	Co-evol. programming
1	91.8309	77.6752
2	65.3700	97.8773
3	93.9287	93.9287
4	93.9933	93.9933

and in both $n = 4$ and $n = 20$, for all the parameter sets used (which were described in the previous section). We used a sample of 300 simulation runs for each parameter set and model, for hypothesis testing. The hypothesis $H_0 : \bar{Q} = q_{Nash}$ was accepted for $a = .05$ in all cases. On the other hand, the hypotheses $H_0 : q_i = q_{Nash}$, were rejected for all players in all models, when the probability of rejection the hypothesis, under the assumption it is correct, was $a = .05$. There was not a single Nash Equilibrium game played, in any of the simulations of the two individual - learning algorithms.

In the social - learning versions of the two algorithms, both the hypotheses $H_0 : \bar{Q} = q_{Nash}$, and $H_0 : q_i = q_{Nash}$ were accepted for $a = .05$, for all models and parameters sets. We used a sample of 300 different simulations for every parameter set, in those cases, as well.

Notice that the all players' quantities have the same mean values (eq. (15)). Mean values of the individual players' quantities on table 2.

Table 2. mean values of the individual players' quantities for $pop = 40$, $p_{cr} = 1$, $p_{mut} = 0.00025$, $T = 10,000$ generations

Player	Social Vriend's alg.	Social Co-evol. prog.	Individual Vriend's alg.	Individual Co-evol. prog.
1	86.9991	87.0062	93.7536	97.4890
2	86.9905	87.0089	98.4055	74.9728
3	86.9994	87.0103	89.4122	82.4704
4	87.0046	86.9978	64.6146	90.4242

On the issue of establishing NE in -some- of the games played and reaching the Nash State (all chromosomes of every population equals the chromosome corresponding to the NE quantity) there are two alternative results. For one subset of the parameters set, the social - learning algorithms managed to reach the NE state and in a significant subset of the games played, all players used the NE strategy.

In the cases where mutation probability was too large, the "Nash" chromosomes were altered significantly and therefore the populations couldn't converge to the NE state (within the given iterations). On the other hand, when the mutation probability was low the number of iterations was not enough to have convergence. A larger population, requires more generations to converge to the "NE state" as well. Apparently, the Nash state s_0 has greater than zero frequency in the simulations

that reach it. The estimated time needed to reach Nash State (in generations), to return to it again after departing from it, and the percentage of total games played that were played on NE, are presented on table 3[1].

Table 3. Percentage of the total games in Nash Equilibrium

Model	Algorithm	pop	p_{mut}	T	Gen NE	Ret Time	NE Games
4-Linear	Vriend	30	.001	10,000	3,749.12	3.83	5.54
4-Linear	Co-evol	40	.0005	10,000	2,601.73	6.97	73.82
20-Linear	Vriend	20	.0005	20,000	2,712.45	6.83	88.98
20-Linear	Co-evol	20	.0001	20,000	2,321.32	6.53	85.64
4-poly	Vriend	40	.00025	10,000	2,483.58	3.55	83.70
4-poly	Co-evol	40	.0005	10,000	2,067.72	8.77	60.45
20-poly	Vriend	20	.0005	20,000	2,781.24	9.58	67.60
20-poly	Co-evol	20	.0005	50,000	2,297.72	,6.63	83.94
4-radic	Vriend	40	.00075	10,000	2,171.32	4.41	81.73
4-radic	Co-evol	40	.0005	10,000	2,917.92	5.83	73.69
20-radic	Vriend	20	.0005	20,000	2,136.31	7.87	75.34
20-radic	Co-evol	20	.0005	20,000	2,045.81	7.07	79.58

6 Conclusions

We have seen that the original individual - learning versions of the multi - population algorithms do not lead to convergence of the individual players' choices, at the Nash Equilibrium quantity. On the contrary, the "socialized" versions introduced here, accomplish that goal and, for a given set of parameters, establish a very frequent Nash State, making games with NE quite frequent as well, during the course of the simulations. The statistical tests employed, proved that the expected quantities chosen by players converge to the NE in the social - learning versions while that convergence cannot be achieved at the individual - learning versions of the two algorithms. Therefore it can be argued that the learning process is qualitatively better in the case of social learning. The ability of the players to take into consideration their opponents strategies, when they update theirs, and base their new choices at the totality of ideas that were used at the previous period (as in [1]), forces the strategies into consideration to converge to each other and to converge to the NE strategy as well. Of course this option would not be possible, if the profit functions of the individual players were not the same, or, to state that condition in an equivalent way, if there were no symmetry at the cost functions. If the cost functions are symmetric, a player can take note of its opponents realized strategies in the course of play, and use

[1] $GenNE$ = Average number of Generations needed to reach s_0, starting from populations having all chromosomes equal to the opposite chromosome of the NE chromosome, in the 300 simulations. $RetTime$ = Interarrival Times of s_0(average number of generations needed to return to s_0) in the 300 simulations. $NEGames$ = Percentage of games played that were NE in the 300 simulations.

them as they are when he updates his ideas, since the effect of these strategies at his individual profit, will be the same. Therefore the inadequate learning process of the individually based learning can be perfected, at the symmetric case. One should note that the convergence to almost identical values displayed in the representative cases of the previous section, holds for any parameter set used in all the models presented in this paper.

References

1. Alkemade, F., La Poutre, H., Amman, H.: On Social Learning and Robust Evolutionary Algorithm Design in the Cournot Oligopoly Game. Comput. Intell. 23, 162–175 (2007)
2. Alos-Ferrer, C., Ania, A.: The Evolutionary Stability of Perfectly Competitive Behavior. Econ. Theor. 26, 497–516 (2005)
3. Arifovic, J.: Genetic Algorithm Learning and the Cobweb Model. J. Econ. Dynam. Contr. 18, 3–28 (1994)
4. Basawa, I.V., Rao, P.: Statistical Inference for Stochastic Processes. Academic Press, London (1980)
5. Dawid, H., Kopel, M.: On Economic Applications of the Genetic Algorithm: A Model of the Cobweb Type. J. Evol. Econ. 8, 297–315 (1998)
6. Dubey, P., Haimanko, O., Zapechelnyuk, A.: Strategic Complements and Subtitutes and Potential Games. Game Econ. Behav. 54, 77–94 (2006)
7. Franke, R.: Coevolution and Stable Adjustments in the Cobweb Model. J. Evol. Econ. 8, 383–406 (1998)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison - Wesley, Reading (1989)
9. Kemeny, J., Snell, J.: Finite Markov Chains. D.Van Nostrand Company Inc., Princeton (1960)
10. Price, T.C.: Using Co-Evolutionary Programming to Simulate Strategic Behavior in Markets. J. Evol. Econ. 7, 219–254 (1997)
11. Protopapas, M., Kosmatopoulos, E.: Two genetic algorithms yielding Nash Equilibrium in Symmetric Cournot Games. COMISEF Working Paper Series, WPS-04 (2008)
12. Riechmann, T.: Learning and Behavioral Stability. J. Evol. Econ. 9, 225–242 (1999)
13. Riechmann, T.: Genetic Algorithm Learning and Evolutionary Games. J. Econ. Dynam. Contr. 25, 1019–1037 (2001)
14. Son, Y.S., Baldick, R.: Hybrid Coevolutionary Programming for Nash Equilibrium Search in Games with Local Optima. IEEE Trans. Evol. Comput. 8, 305–315 (2004)
15. Vallee, T., Yildizoglou, M.: Convergence in Finite Cournot Oligopoly with Social and Individual Learning. Working Papers of GREThA, 2007-07 (2007), GREThA, http://www.gretha.fr (accessed November 10, 2007)
16. Vriend, N.: An Illustration of the Essential Difference between Individual and Social Learning, and its Consequences for Computational Analyses. J. Econ. Dynam. Contr. 24, 1–19 (2000)

Multiple Overlapping Tiles for Contextual Monte Carlo Tree Search

Arpad Rimmel and Fabien Teytaud

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud), bat 490 Univ. Paris-Sud
91405 Orsay, France

Abstract. Monte Carlo Tree Search is a recent algorithm that achieves more and more successes in various domains. We propose an improvement of the Monte Carlo part of the algorithm by modifying the simulations depending on the context. The modification is based on a reward function learned on a tiling of the space of Monte Carlo simulations. The tiling is done by regrouping the Monte Carlo simulations where two moves have been selected by one player. We show that it is very efficient by experimenting on the game of Havannah.

1 Introduction

Monte Carlo Tree Search (MCTS) [1] is a recent algorithm for taking decisions in a discrete, observable, uncertain environment with finite horizon that can be described as a reinforcement learning algorithm. This algorithm is particularly interesting when the number of states is huge. In this case, classical algorithms like Minimax and Alphabeta [2], for two-player games, and Dynamic Programming [3], for one-player games, are too time-consuming or not efficient. As MCTS explores only a small relevant part of the whole problem, this allows it to obtain good performance in such situations. This algorithm achieved particularly good results in two-player games like computer Go or Havannah. But this algorithm was also successfully applied on one-player problems like the automatic generation of libraries for linear transforms [4] or active learning [5].

The use of Monte Carlo simulations to evaluate a situation is an advantage of the MCTS algorithm; it gives an estimation without any knowledge of the domain. However, it can also be a limitation. The underlying assumption is that decisions taken by an expert are uniformly distributed in the whole space of decisions. This is not true in most of the cases. In order to address this problem, one can add expert knowledge in the Monte Carlo simulations as proposed in [6]. However, this solution is application-dependent and limited because all the different situations have to be treated independently.

In this paper, we present a first step to solve this problem in a generic way. We introduce a modification of the Monte Carlo simulations that allows them to be automatically modified depending on the context: Contextual Monte Carlo (CMC) simulations. We show that it improves the performance for the game of Havannah. In order to do that, we learn the reward function on a tiling of the

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 201–210, 2010.

space of Monte Carlo simulations and use this function to modify the following simulations. The idea is to group simulations where two particular actions have been selected by the same player. Then, we learn the average reward on those sets. And finally, we try to reach simulations from sets associated to a high reward. This modification is generic and can be applied to two-player games as well as one-player games. To the extent of our knowledge, this is the first time a generic and automatic way of adapting the Monte Carlo simulations in the MCTS algorithm has been proposed.

We first present reinforcement learning, the principle of the Monte Carlo Tree Search algorithm and the principle of tiling. Then, we introduced those new simulations: CMC simulations. Finally, we present the experiments and conclude.

2 Value-Based Reinforcement Learning

In a reinforcement learning problem, an agent will choose *actions* in an environment described by *states* with the objective of maximizing a long-term *reward*. The agent will act based on his previous trials (*exploitation*) and try new choices (*exploration*).

Let S be a set of states. Let A be a set of actions. Let $R \subset \mathbb{R}$ be a set of rewards.

At each time $t \in 1, ..., T$, the current state $s_t \in S$ is known. After an action $a_t \in A$ is chosen, the environment returns the new state s_{t+1} and the reward $r_{t+1} \in R$.

The goal is to find a policy function $\pi : S \to A$ that maximizes the cumulative reward R_t for each t:

$$R_t = \sum_{k=t+1}^{T} r_k$$

In value-based reinforcement learning, an intermediate *value function* is learned to compute the policy. The value function $V^\pi(s)$ is the expected cumulative reward starting state s and following the policy π thereafter.

$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s]$$

This function is not known and will be empirically evaluated: $\hat{V}^\pi$.

While there is some time left, the algorithm will iterate two different steps:

- *utilization* of the empirical estimation of the value function. $\hat{V}^\pi$ is used in order to choose the actions (often based on a compromise between exploration and exploitation).
- *update* of the empirical estimation of the value function. $\hat{V}^\pi$ is modified based on the rewards obtained.

This is known as the *policy iteration* process [7].

The utilization and update of $\hat{V}^\pi$ can be done by different algorithms. For example, [8] propose the $TD(\lambda)$ algorithm in order to do the update. A classical utilization of $\hat{V}^\pi$ is the ϵ-greedy algorithm.

3 Monte Carlo Tree Search

The principle of MCTS is to construct a highly unbalanced tree representing the future by using a bandit formula and to combine it with Monte Carlo simulations to evaluate the leaves.

3.1 Bandits

A classical k-armed bandit problem is defined as follows:

- A finite set $J = \{1, \dots, k\}$ of arms is given.
- Each arm $j \in J$ is associated to an unknown random variable X_j with an unknown expectation μ_j.
- At each time step $t \in \{1, 2, \dots\}$:
  - the algorithm chooses $j_t \in J$ depending on $(j_1, \dots, j_{t-1})$ and $(r_1, \dots, r_{t-1})$.
  - Each time an arm j is selected, the bandit gives a reward r_t, which is a realization of X_{j_t}.

The goal of the problem is to minimize the so-called *regret*: the loss due to the fact that the algorithm will not always chose the best arm.

Let $T_j(n)$ the number of times an arm has been selected during the first n steps. The *regret* after n steps is defined by

$$\mu^* n - \sum_{j=1}^{k} \mu_j \mathbb{E}[T_j(n)] \text{ where } \mu^* = \max_{1 \leq i \leq k} \mu_i$$

[9] achieve a logarithmic regret (it has been proved that this is the best regret obtainable in [10]) uniformly over time with the following algorithm: first, tries one time each arm; then, at each step, selects the arm j that maximizes

$$\bar{x_j} + \sqrt{\frac{2ln(n)}{n_j}} \tag{1}$$

$\bar{x_j}$ is the average reward for the arm j.
n_j is the number of times the arm j has been selected so far.
n is the overall number of trials so far.

This formula consists in choosing at each step the arm that has the highest upper confidence bound. It is called the UCB formula.

3.2 Monte Carlo Tree Search

The MCTS algorithm constructs in memory a subtree $\hat{T}$ of the global tree T representing the problem in its whole (see algorithm 1 (left) and Fig. 1 (left)).

The construction of the tree is done by the repetition while there is some time left of 3 successive steps: *descent, evaluation, growth*.

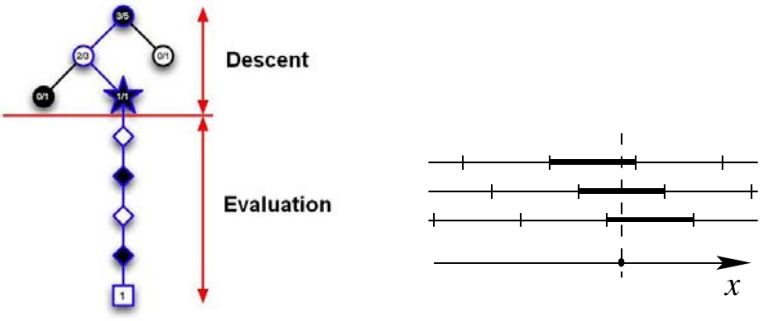


Fig. 1. Left. Illustration of the Monte Carlo Tree Search algorithm from a presentation of Sylvain Gelly. **Right.**Illustration of 3 overlapping tilings from the article [11].

Descent. The descent in $\hat{T}$ is done by considering that taking decision is a k-armed bandit problem. We use the formula 1 to solve this problem. In order to do that, we suppose that the necessary information is stored for each node. Once a new node has been reached, we just repeat the same principle until we reached a situation S outside of $\hat{T}$.

Evaluation. Now that we have reached S and that we are outside of $\hat{T}$, there is no more information available to take a decision. As we are not at a leaf of T, we can not directly evaluate S. Instead, we use a Monte Carlo simulation (taking decisions uniformly until a final state is reached) to have a value for S.

Growth. We add the node S to $\hat{T}$. We update the information of S and of all the situations encountered during the descent with the value obtained with the Monte Carlo evaluation.

3.3 Monte Carlo Tree Search as a Reinforcement Learning Algorithm

The tree representing the problem solved by MCTS can be described as a reinforcement learning problem with the following correspondence: states $\sim$ nodes of the tree, actions $\sim$ branches of the tree, rewards $\sim$ results at the terminal nodes of the tree.

The MCTS algorithm is a value-based reinforcement learning algorithm with a UCB policy. The value $\hat{V}^{UCB}(s)$ is stored in the node corresponding to the state s. It corresponds to the average reward for the situation s so far.

The *utilization* part of the algorithm is defined as follows: the action a chosen in the state s is selected according to

$$\begin{cases} argmax_a(\hat{V}^{UCB}(s_a) + \sqrt{\frac{2ln(n_s)}{n_{s_a}}}) & \text{if } s \in \hat{T} \\ mc(s) & \text{otherwise} \end{cases} \tag{2}$$

s_a is the situation reached from s after choosing the action a.

n_{s_a} is the number of times the action a has been selected so far from the situation s.

n_s is the overall number of trials so far for situation s.

$mc(s)$ returns an action uniformly selected among all the possible actions from the state s.

When s is in $\hat{T}$, the action is chosen according to the UCB formula 1 (descent part). When s is outside of $\hat{T}$, the action is chosen uniformly among the possible actions (evaluation part).

The *update* part of the reinforcement learning algorithm is done after a final state is reached and a new node has been added to $\hat{T}$. The reward is the value r associated to the final state. For all states $s \in \hat{T}$ that were reached from the initial state to the final state

$$\hat{V}^{UCB}(s) \leftarrow \hat{V}^{UCB}(s) + \frac{r}{n_s}$$

4 Tile Coding

When the number of states is very large or even infinite in the case of continuous parameters, it is necessary to use a *function approximation* to learn the value function.

In *tile coding* (see [7]), the space D is divided into *tiles*. Such a partition is called a *tiling*. It is possible to use several overlapping tilings. A weight is associated by the user to each tile. The value of a point is given by the sum of the weight of all the tiles in which the point is included. A representation of 3 overlapping tilings for a problem with one continuous dimension is given on Fig. 1 (right).

Tile coding will lead to a piecewise constant approximation of the value function:

$$\forall p \in D, \exists z \in \mathbb{R} \text{ such that, } \forall p' \in D \wedge distance(p, p') < z, \hat{V}^{\pi}(p) = \hat{V}^{\pi}(p')$$

5 Contextual Monte Carlo

In this section, we present how we learn the reward function on the space of Monte Carlo simulations by defining a tiling on this space. Then, we explain how we use this function to improve the following Monte Carlo simulations.

5.1 A New Tiling on the Space of Monte Carlo Simulations

We consider a planning problem, the goal is to maximize the reward. We consider that a Monte Carlo Tree Search algorithm is used to solve this problem. Let G be the set of the possible actions. We focus on the space of Monte Carlo simulations E_{MC}. A Monte Carlo simulation is the sequence of moves from outside the tree $\hat{T}$ until a final state. Each Monte Carlo simulation is therefore associated to a

reward. We define the tiles $L(a_1, a_2)$ on E_{MC} where $(a_1, a_2) \in G^2$. $L(a_1, a_2)$ is composed of all the simulations containing a_1 and a_2 and where a_1 and a_2 has been selected by one player P.

$$L = \{\{\text{sim } s \text{ such that } a_1 \in s \wedge a_2 \in s \wedge P_s(a_1) \wedge P_s(a_2)\}; (a_1, a_2) \in G^2\} \quad (3)$$

We define $\hat{V}_{CMC}$: the empirical reward function based on L. In order to learn the value $\hat{V}_{CMC}(a_1, a_2)$, each time that a simulation s is rewarded with a value r, we update the values for each tiles containing s.

For each $L(a_1, a_2)$ such that $s \in L(a_1, a_2)$,

$$\hat{V}^{CMC}(a_1, a_2) \leftarrow \hat{V}^{CMC}(a_1, a_2) + \frac{r}{n_{CMC}(a_1, a_2)}$$

$n_{CMC}(a_1, a_2)$ is the number of times a simulation in $L(a_1, a_2)$ has been played.

$\hat{V}^{CMC}(a_1, a_2)$ corresponds to the estimated reward for any simulation in which two particular actions have been selected. If this value is high, it means that each time the player manages to play a_1 and a_2 in a simulation, there is a high chance that the simulation will give a high reward for that player.

5.2 Improving Monte Carlo Simulations

We focus on tiles where the estimated reward is high (superior to a user-defined threshold B). The policy should try to reach simulations in those tiles. In order to do that, if a tile associated with two actions a_1 and a_2 and with an estimated value inferior to B exists and if one player previously selected one of the actions, we will then select the other. In fact, if several such tiles exist, we will select the action that will lead to the simulation from the tile with the highest average reward.

As the policy for situations in $\hat{T}$ is already efficient, we modify the policy for situations outside $\hat{T}$.

The utilization part previously defined in Eq. 2 is now defined as follows: The action a chosen in the state s is selected according to

$$\begin{cases} argmax_a(\hat{V}^{UCB}(s_a) + \sqrt{\frac{2ln(n_s)}{n_{s_a}}}) & \text{if } s \in \hat{T} \\ \begin{cases} cmc(s) \text{ if } random() < prob \\ mc(s) \quad \text{otherwise} \end{cases} & \text{otherwise} \end{cases} \quad (4)$$

$prob$ is a parameter between 0 and 1. It corresponds to the probability of choosing $cmc(s)$ instead of $mc(s)$. $random()$ is a random number generated between 0 and 1. $cmc(s)$ is defined as follows:

$$cmc(s) = argmax_{a, a \in E_s}(S(\hat{V}^{CMC}(a, b)))$$

E_s is the set of the possible actions in the state s. b is the previous move played by the same player. S is a threshold function with a threshold at B, formally defined as follows:

$$S(x) = \begin{cases} x & \text{if } x > B \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In order to keep the diversity in Monte Carlo simulation (the importance of diversity is discussed in [6]), we apply this modification with a certain probability *prob*. This probability is defined by the user.

The resulting algorithm is given in algorithm 1 (right).

Algorithm 1. Left. MCTS(s) **Right.** CMCTS(s) //s a situation

Initialization of $\hat{T}$, $\hat{V}^{UCB}$, n
while there is some time left **do**
  $s' = s$
  Initialization of *game*
  //*DESCENT*//
  while s' in $\hat{T}$ and s' not terminal **do**
    $s' =$ reachable situation chosen according to the UCB formula (1)
    $game = game + s'$
  end while
  $S = s'$
  //*EVALUATION*//
  while s' is not terminal **do**
    $s' = mc(s')$
  end while
  $r = result(s')$
  //*GROWTH*//
  $\hat{T} = \hat{T} + S$
  for each s in *game* **do**
    $n_s \leftarrow n_s + 1$
    $\hat{V}^{UCB}(s) \leftarrow \hat{V}^{UCB}(s) + \frac{r}{n_s}$
  end for
end while

Initialization of $\hat{T}$, $\hat{V}^{UCB}$, n, $\hat{V}^{CMC}$, n^{CMC}
while there is some time left **do**
  $s' = s$
  Initialization of *game*,*gamemc*
  //*DESCENT*//
  while s' in $\hat{T}$ and s' not terminal **do**
    $s' =$ reachable situation chosen according to the UCB formula (1)
    $game = game + s'$
  end while
  $S = s'$
  //*EVALUATION*//
  while s' is not terminal **do**
    if $random() < prob$ **then**
      $s' = cmc(s')$
    else
      $s' = mc(s')$
    end if
    $gamemc \leftarrow gamemc + s'$
  end while
  $r = result(s')$
  //*GROWTH*//
  $\hat{T} = \hat{T} + S$
  for each s in *game* **do**
    $n_s \leftarrow n_s + 1$
    $\hat{V}^{UCB}(s) \leftarrow \hat{V}^{UCB}(s) + \frac{r}{n_s}$
  end for
  for each $(P(a_1), P(a_2))$ in s', P being one player **do**
    $n^{CMC}(a_1, a_2) \leftarrow n^{CMC}(a_1, a_2) + 1$
    $\hat{V}^{CMC}(a_1, a_2) \leftarrow \hat{V}^{CMC}(a_1, a_2) + \frac{r}{n^{CMC}(a_1, a_2)}$
  end for
end while

6 Experiments

We have tested the effect of contextual Monte Carlo simulations on the game of Havannah. In this section, we first describe the game of Havannah and then give our results.

6.1 Havannah

Havannah is a two-player board game recently introduced in the community of computer game [12]. Invented by Christian Freeling, the game of Havannah is played on an hexagonal board with hexagonal locations, and different board sizes (size of 8 or 10 is usual). The rules are really simple. White player starts, and after that each player plays alternately by putting a stone in an empty location. If there is no any empty location free, and if no player has won yet, then the game is a draw. To win, a player has to realize :

- a ring, which is a loop around one or more cells (empty or not, occupied by black or white stones).
- a bridge, which is a continuous string of stones connecting to one of the six corners to another one.
- a fork, which is a continuous string of stones linking three edges of the board (corner locations are not considered as belonging to the edges).

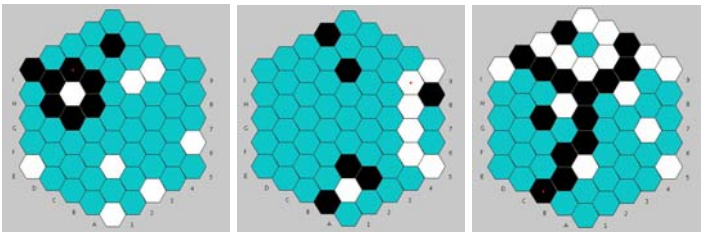


Fig. 2. Three finished games: a ring (a loop, by black), a bridge (linking two corners, by white) and a fork (linking three edges, by black)

On Fig. 2, we present these three ways to win a game.

The game of Havannah is known as hard for computers for different reasons. First, there is a large action space, for instance, in size 10, there are 271 possible moves for the first player. Second, there is no pruning rule for reducing the number of possible moves. Another important reason is that, there is no natural evaluation function. A such function is really useful, in the sense that, it gives a very good evaluation of a position, as, for instance in chess. And finally, a last reason is the lack of patterns in the game of Havannah. A pattern is an expert knowledge which give information on a move or a position. For instance, in chess, it is always good having his king in a safe place, therefore, a pattern could be castling if it is possible.

6.2 Results

In this section we experiment our Havannah program with the CMC improvement against the same player without this improvement. The reward is 1 (if the situation is a won game) or 0 (if the situation is a loss). The experiments are done with 1000 simulations per move for each player.

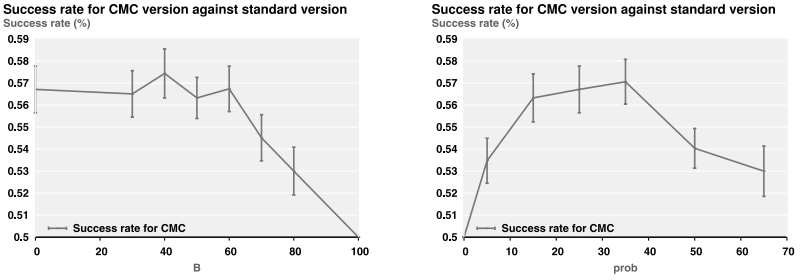


Fig. 3. Left. Winning percentage for the CMC version against the basic version with $prob = 35$. **Right.**Winning percentage for the CMC version against the basic version with $B = 0$.

We study the impact of the two parameters *prob* and B. *prob* defines the percentage of time our modification will be applied (see Algo. 1). B defines which tiles we want to reach (see Eq. 5). The results are given on Fig. 3 and are commented below.

First, we see that the utilization of CMC is efficient. It leads to 57% of victory against the base version for $prob = 35$ and $B = 0$.

On Fig. 3 (left), we show the effect of changing B for a fixed value of $prob = 75\%$. The winning percentage starts at 56% for $B = 0$ and is stable until $B = 60$ where it starts going down to 50% for $B = 100$. When B is too high, CMC is used less often and therefor the results are worse. When B is low, it means that we will select the best tile even if all the possible tiles have a bad average reward. It seems that this is never worst than playing randomly. In the following, we use $B = 0$.

On Fig. 3 (right), we modify the value of *prob* while keeping B fixed. When *prob* is too high, the diversity of the Monte Carlo simulations is not preserved and the results are worse. On the other hand, if *prob* is too low, the modification has not enough effect. There is a compromise between this two properties.

7 Conclusion

We presented a domain-independent improvement of the Monte Carlo Tree Search algorithm. This was done by modifying the MC simulations by using a reward function learned on a tiling of the space of MC simulations: CMC. To the extent of our knowledge, this is the first time that an automatic modification of the Monte Carlo has been proposed.

It achieves very good results for the game of Havannah with a winning percentage of 57% against the version without CMC.

It is, for the moment, tested only in the case of one example of two-player game. An immediate perspective of this work is to experiment CMC on other problems.

It is possible to apply an infinite amount of tilings to the space of Monte Carlo simulations. We proposed a successful specific one but others can surely be found as we used only a small part of the information contained in this space. In the future, we intend to give a formal description of the learning in the space of Monte Carlo simulations.

References

1. Kocsis, L., Szepesvari, C.: Bandit-based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
2. Pearl, J.: Heuristics. Intelligent search strategies for computer problem solving. Addison-Wesley, Reading (1984)
3. Bertsekas, D.P.: Neuro-dynamic programming. In: Encyclopedia of Optimization, pp. 2555–2560 (2009)
4. De Mesmay, F., Rimmel, A., Voronenko, Y., Püschel, M.: Bandit-Based Optimization on Graphs with Application to Library Performance Tuning. In: International Conference on Machine Learning, Montréal Canada (2009)
5. Rolet, P., Sebag, M., Teytaud, O.: Optimal active learning through billiards and upper confidence trees in continous domains. In: Proceedings of the European Conference on Machine Learning (2009)
6. Chaslot, G., Fiter, C., Hoock, J.B., Rimmel, A., Teytaud, O.: Adding expert knowledge and exploration in Monte-Carlo Tree Search. In: Advances in Computer Games, Pamplona Espagne. Springer, Heidelberg (2009)
7. Sutton, R., Barto, A.: Reinforcement learning: An introduction. MIT Press, Cambridge (1998)
8. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning, 9–44 (1988)
9. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2/3), 235–256 (2002)
10. Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6, 4–22 (1985)
11. Sherstov, E.A., Stone, P.: Function approximation via tile coding: Automating parameter choice. In: Zucker, J.-D., Saitta, L. (eds.) SARA 2005. LNCS (LNAI), vol. 3607, pp. 194–205. Springer, Heidelberg (2005)
12. Teytaud, F., Teytaud, O.: Creating an Upper-Confidence-Tree program for Havannah. In: Advances in Computer Games 12, Pamplona Espagne (2009)

A CNN Based Algorithm for the Automated Segmentation of Multiple Sclerosis Lesions

Eleonora Bilotta[1], Antonio Cerasa[2], Pietro Pantano[1], Aldo Quattrone[2], Andrea Staino[1], and Francesca Stramandinoli[1]

[1] Evolutionary Systems Group, University of Calabria, 87036 Arcavacata di Rende, Cosenza, Italy
{bilotta,piepa}@unical.it, {andreastaino,francescastramandinoli}@gmail.com
[2] Neuroimaging Research Unit, Institute of Neurological Sciences, National Research Council, 88100 Catanzaro, Italy
{a.cerasa,a.quattrone}@isn.cnr.it

Abstract. We describe a new application based on genetic algorithms (GAs) that evolves a Cellular Neural Network (CNN) capable to automatically determine the lesion load in multiple sclerosis (MS) patients from Magnetic Resonance Images (MRI). In particular, it seeks to identify in MRI brain areas affected by lesions, whose presence is revealed by areas of lighter color than the healthy brain tissue. In the first step of the experiment, the CNN has been evolved to achieve better performances for the analysis of MRI. Then, the algorithm was run on a data set of 11 patients; for each one 24 slices, each with a resolution of 256×256 pixels, were acquired. The results show that the application is efficient in detecting MS lesions. Furthermore, the increased accuracy of the system, in comparison with other approaches, already implemented in the literature, greatly improves the diagnosis for this disease.

Keywords: Cellular Neural Networks, Genetic Algorithms, Automated Magnetic Resonance Imaging Analysis, Multiple Sclerosis lesion load.

1 Introduction

Cellular Neural Networks (CNNs), first introduced by Leon O. Chua and Lin Yang [3] in 1988, are an array of nonlinear programmable analog processors, called cells, that perform parallel computation. Each cell is a dynamical system whose state evolves in time, according to a specific mathematical model, and whose output is a nonlinear function of the state. Unlike artificial neural networks, in a CNN interconnections among cells are local, that is each processing unit directly interacts only with the neighboring cells, located within a prescribed sphere of influence. For image processing purpose, the most usual CNN architecture is a regular two dimensional grid. Given a CNN of $M \times N$ cells, the neighborhood $S_{ij}(r)$ of radius $r \geq 0$ for the cell C_{ij} is the set of cells satisfying the following property:

$$S_{ij}(r) = \{C_{kl} : \max(|k-i|, |l-j|) \leq r\} \quad 1 \leq k \leq M,\ 1 \leq l \leq N \qquad (1)$$

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 211–220, 2010.

In the original model [3], each CNN cell is a simple nonlinear analog circuit (Fig. 1), composed of a linear capacitor, an independent current source, an independent voltage source, two linear resistors and at most $2m$ linear voltage-controlled current sources, m being the number of neighbors cells of the considered unit. The voltage $v_{xij}(t)$ across the capacitor is called the state of the cell C_{ij}, while v_{uij} and $v_{yij}(t)$ represent the input and the output respectively. The characteristics of the generators $I_{xy}(i,j;k,l;t)$ and $I_{xu}(i,j;k,l)$ are defined as:

$$I_{xy}(i,j;k,l;t) = A(i,j;k,l)v_{y_{kl}}(t), \quad I_{xu}(i,j;k,l) = B(i,j;k,l)v_{u_{kl}} \tag{2}$$

By setting the coupling parameters $A(i,j;k,l)$ and $B(i,j;k,l)$, it is possible to control the strength of interactions between cells. The output $v_{y_{ij}}(t)$ is determined by the nonlinear voltage controlled current source I_{yx} that is the only nonlinear element of the cell. It is characterized by the following equation:

$$I_{yx} = \frac{1}{R_y} f\left(v_{x_{ij}}(t)\right) \tag{3}$$

where f is the characteristic function of the nonlinear controlled current source, defined as:

$$f\left(v_{ij}(t)\right) = \frac{1}{2}\left(|v_{x_{ij}}(t)+1| - |v_{x_{ij}}(t)-1|\right) \tag{4}$$

Using the Kirchhoff laws, the state of a CNN cell can be described by the following nonlinear differential equation:

$$C\,\dot{v}_{x_{ij}}(t) = -\frac{1}{R_x}v_{x_{ij}}(t) + z + \sum_{C_{kl}\in S_{ij}(r)} \left(A(i,j;k,l)f\left(v_{x_{kl}}(t)\right) + B(i,j;k,l)v_{u_{kl}}\right) \tag{5}$$

where f holds the nonlinearity. Therefore, given input, initial state for each cell C_{ij} such that $1 \le i \le M$, $1 \le j \le N$, and boundary conditions, the dynamics of a two-dimensional standard CNN are uniquely specified by the synaptic weights between a cell and its neighbors. These parameters, together with a bias value z, define a CNN template that can be expressed in the form $\{A(i,j;k,l),\ B(i,j;k,l),\ z\}$. The process performed by the system on the input image is fully defined by the set of coefficients in the CNN template. An evolutionary approach can be used in order to find a template that allows to obtain a desired process.

In [2], CNNs are proposed as a parallel computing paradigm especially suited for processing analog array signals, with important applications in image processing, pattern recognition, numerical solution of PDEs and investigation of nonlinear phenomena. CNNs have been successfully applied in various image processing applications, especially because of the high pay-off offered by the CNN based architectures [1]. Neuroimaging is one of the most important area in which CNNs were also used in order to support medical diagnosis, both with magnetic resonance imaging (MRI) and computed tomography (CT) [5,8]. In the last years, there has been increased interest in developing novel techniques for automated multiple sclerosis (MS) lesions segmentation. MS is a demyelinising disease of the central nervous system that leads to inflammatory pathology. MS

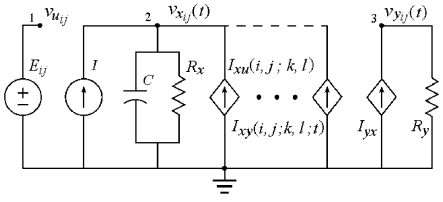


Fig. 1. Original CNN cell model

pathology is primarily expressed as focal lesions in the white matter of the brain. Because of its superior contrast, MRI is the modality of choice for clinical evaluation of MS. Generally, manual delineation of MS lesions is a time-consuming task, as three-dimensional information, from several MR contrasts, must be integrated. In [5], a CNN based approach to classify MR images with respect to the presence or absence of mesial temporal sclerosis has been proposed, using a genetic algorithm-based learning procedure in order to optimize the networks' parameters, concerning the assigned classification task. By developing a new and efficient CNN based approach of MR images processing and by using a genetic algorithm which improves the technique developed in [5], this paper presents a fully automated method for the segmentation of MS lesions. The paper is organized as follows. After the introduction, the second section deals about the GA formal aspects to evolve the CNN templates. The third section reports about the CNN simulation to test the system's implementation. The fourth and the fifth sections present the experiments we have performed and some results. The final remarks close the work.

2 GA Methods for Evolving CNN

From the work of J.H. Holland in 1975, Genetic Algorithms (GAs) are computational bio-inspired methods for solving problems. To evaluate the performance of each individual in relation to the problem, it is possible to define an appropriate fitness function, which quantitatively measures the performance of each individual, in a given generation, for all the generations [1]. The standard method for developing a GA is to choose a genetic representation, a fitness function and then it proceeds with the following steps:

1. Generating a random number of strings (initial population), that encode possible solutions to the problem.
2. Decoding of the genotypes of the population and assessment of each individual (phenotype), according to the fitness function.
3. If the current population contains a satisfactory solution, the algorithm stops.
4. If the system doesn't find a "good" solution, a new evolution starts, generating a new population of individuals, by applying the operators of selection, crossover and mutation.

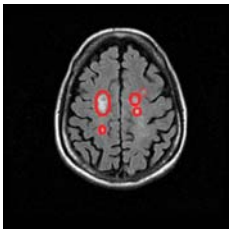


Fig. 2. Marked areas corresponding to MS lesions in the white matter of the brain. Lesions are brighter than other tissues on MRI.

The process continues with the evaluation of new individuals through the fitness function and continues cyclically in this manner until a satisfactory solution to a given problem is obtained. GAs and, more generally, evolutionary computing, have been successfully applied to image processing tasks related to medical images classification [10]. In this paper, genetic algorithms have been applied in order to evolve a CNN, capable of detecting MS lesions from MRI. The main issue is to develop a CNN algorithm for automated segmentation of MS lesions, whose presence is revealed by regions in the brain that are brighter than their surroundings (Fig. 2).

In image processing applications, a neighborhood of radius $r = 1$ is commonly used and, in most cases, space-invariant templates are chosen, that is the operators $A(i, j; k, l)$ and $B(i, j; k, l)$ depend only on the relative position of a cell with respect to its neighbors. With such assumption, the whole system is characterized by a 3×3 feedback matrix A, a 3×3 control matrix B and a scalar z and so 19 parameters are needed to "program" a cellular neural network; this means that, once initial state and boundary conditions have been assigned, the operation performed by the CNN on a given input image is determined only by 19 real values that completely define the properties of the network. For our aims, to design a genetic algorithm to search the weights of a standard two-dimensional space invariant CNN, in which each cell has a radius of influence $r = 1$, it is convenient to adopt a representation of templates in vector form. To this end, the 19 parameters that define the triple $\{A, B, z\}$ are arranged in an array consisting of 9 feedback synaptic weights, defining the A matrix, 9 control synaptic weights, defining the B matrix, and the threshold z (Fig. 3). These 19 coefficients represent a gene for the CNN, which is associated with a particular function performed by the network. The genetic algorithm has been designed to get template to be used for image processing applications. For this reason, we chose to impose that the matrices A and B are symmetric with respect to the central element, respectively.

In this way, we set the conditions for the stability of the CNN, provided in the complete stability theorem [2], which ensures the convergence of the network. It also reduces the computational load of the algorithmic search, since it is necessary to determine only 11 coefficients, 5 belonging to the matrix A, 5 to the

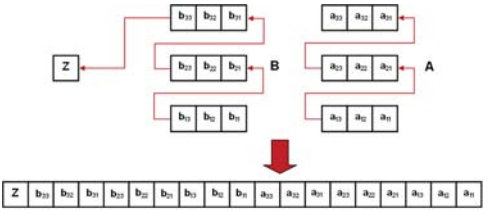


Fig. 3. Representation of a CNN template in vector form

matrix B and one corresponding to the threshold z. Each genotype is represented by a vector G of 11 elements:

$$G = [a_{11} \quad a_{12} \quad a_{13} \quad a_{21} \quad a_{22} \quad b_{11} \quad b_{12} \quad b_{13} \quad b_{21} \quad b_{22} \quad z] \tag{6}$$

while the corresponding phenotype is the actual cellular neural network, configured by using the parameters in the genotype. The key step in the genotype-to-fenotype mapping is the construction of the CNN template $\{A, B, z\}$ from the elements of a given vector G; this can be easily accomplished by re-arranging genotype coefficients as shown in Fig. 3. To assess the fitness of a CNN gene compared to an assigned problem, we introduce a target image T of $M \times N$ pixels to be used for training the network. Applying the template corresponding to G to the input image to CNN, it generates an image I^G which can be compared with T, through the cost function:

$$diff(G) = \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}^{G} \bigoplus T_{ij} \tag{7}$$

where the operator $\oplus$ denotes the logic *xor* between the element in position (i, j) of the target image and the corresponding pixel in the CNN output. The fitness function for each phenotype CNN^G, then, is evaluated by calculating the number of pixels equal between T and the CNN output:

$$fitness(CNN^G) = M \times N - diff(G) \tag{8}$$

Hence, the fitness measures the number of equal pixels between the target image and that obtained from the CNN simulation. In this way, higher values of fitness are associated with phenotypes corresponding to templates that produce outputs with a high number of pixels, that in turn coincide with the image target.

3 Experiments on the CNN Performance

The algorithm proposed for the segmentation of the MS lesions consists of three principal steps:

(a) (b)

Fig. 4. Input (a) and target (b) image used in the CNN training

– Step 1: Lesions detection and their binarization,
– Step 2: Segmentation of the white matter of the brain,
– Step 3: Lesions extraction and isolation.

Genetic algorithms have been applied to determine the CNN synaptic weights that, for a given MR Image, perform a binarization in such a way that only MS lesions are detected inside the brain area. The subsequent steps have been necessary in order to remove the skull and other unwanted pixels. Because of different shapes and intensity of the lesions, it has been necessary to train the genetic algorithm on images presenting different characteristics; for this reason, the network was evolved using Fig. 4(a) as input and Fig. 4(b) as the corresponding desired output.

The evolution of the CNN has been carried out using CNNSimulator, a software environment for the analysis of CNN dynamics; at each step of the training process, the error function to be minimized by the GA was the number of different pixels between the desired and the actual output of the CNN. In our implementation, we ran an initial random population of 35 individuals, making them evolve for 300 generations; the number of individuals was kept constant during the evolutionary process, weighted roulette wheel selector was used as selection method, mutations and elitism strategies were applied. In order to reduce the computational effort due to the large search space, we chose to constrain the elements of each genotype to be in the range $[-8, 8]$. The GA was conducted as follows: after evaluating the fitness of each phenotype, the elite individual, i.e. the most performant one, has been directly copied in the next generation; a number of single-point crossover operations, depending on the population size, has been performed. In our experiments, we used a crossover percentage of 30%, meaning that the number of genetic crossing over operations has been 0.3 multiplied by the population size. Mutations have been randomly applied in order to prevent trapping into local minima. The elements of the genotypes in the population have been randomly mutated according to a given mutation rate, that is each coefficient had a given probability of being changed by a randomly selected real number that falls in the chosen interval $[-8, 8]$. Using a mutation rate of 0.05, each component had 5% probability of being changed, resulting in 1/20 parameters being mutated on average. Once genetic operators have been applied, a fixed number of genotypes has been selected and moved on the next generation population. Obviously, the selection has been guided by the fitness,

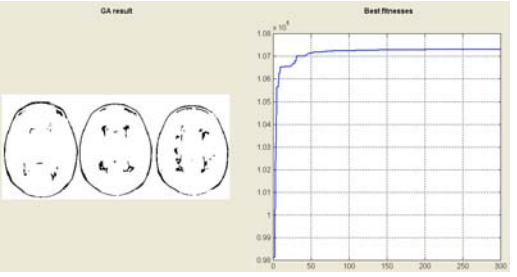


Fig. 5. Evolution of the Cellular Neural Network

i.e. higher probabilities of survival have been associated to phenotypes providing higher fitness values. At the end of the training process, the following template matrices were found:

$$A = \begin{bmatrix} -3.51879 & 3.42019 & -3.48386 \\ 6.47032 & 7.75293 & 6.47032 \\ -3.48386 & 3.42019 & -3.51879 \end{bmatrix} \quad B = \begin{bmatrix} 1.33076 & -3.86887 & 1.53728 \\ -2.30849 & -7.76398 & -2.30849 \\ 1.53728 & -3.86887 & 1.33076 \end{bmatrix} \quad z = -4.81797 \quad (9)$$

Figure 5 shows the output generated by the CNN corresponding to the highest fitness value, together with the fitness achieved by the best individual in each generation.

The removal of the skull and other unwanted features has been achieved by an AND operation between the output of the evolved CNN and the corresponding white matter of the brain, segmented in the second step of the algorithm. It gave the MS lesions in output, while the remaining parts have been removed. We used SPM8 [7] for white matter segmentation, while greyscale image binarization and logic AND operation could be easily performed, by using the templates proposed in the CNN software library [6]. The third step of the proposed algorithm is shown in Fig. 6. Once the pixels corresponding to the lesions have been extracted for each slice of a given MS patient, knowing the voxel size in the acquired MRI sequence, it is possible to perform a quantitative evaluation of the MS total lesion load (TLL). The performances of the process have been quantitatively evaluated by comparing the CNN output and the expert's manual delineation of MS lesions, using the Dice coefficient [4] as a metric. The Dice coefficient D is a statistic measure used for comparing the extent of spatial overlap between two binary images. It is commonly used in reporting performance of segmentation and its values range between 0 (no overlap) and 1 (perfect agreement). In this paper the Dice values, expressed as percentages, are computed as follows:

$$D = \frac{2\left|L^{CNN} \cap L^{G}\right|}{|L^{CNN}| + |L^{G}|} \times 100 \quad (10)$$

where L^{CNN} is the automated segmentation result and L^{G} the manual one. We applied the algorithm to a data set of real MR images acquired from 11 MS patients, for whom 24 slices were taken to cover the whole brain. Each slice

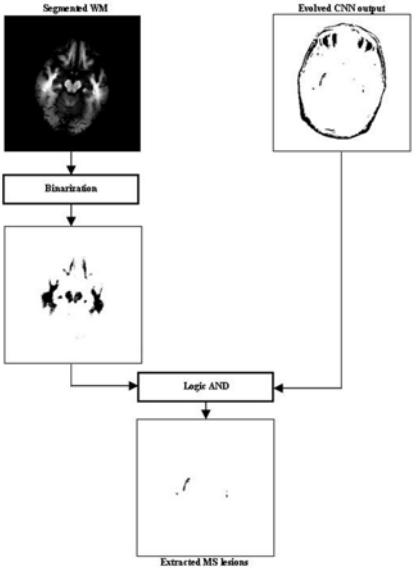


Fig. 6. Skull and unwanted features removal

had a resolution of 256×256 pixels and voxel size is $0.94mm \times 0.94mm \times 5.00mm$. The simulation results showed that the CNN based system is effective in segmenting MS lesions in fast fluid-attenuated inversion-recovery (FLAIR) axial images.

4 Results

The exposed method gives satisfactory results, showing that after the learning process the cellular neural network is capable of detecting MS lesions with different shapes and intensities, even in MRI slices with different contrasts between white and grey matter with respect to the images used during the genetic training process. The vast majority of the lesion load, detected by CNN for the described sample, ranges from $D = 0.7$ to $D = 0.8$. The technique we propose for segmenting white matter lesions in MS is a fully automatic method and does not require manually segmented data; in fact, while semiautomatic methods [9] are highly dependent on the choice of an appropriate threshold to effectively detect lesions (threshold that usually may vary between different slices even for the same patient, thus leading to a time consuming task), our algorithm allows for obtaining the desired output by programming a fully automated strategy on the entire data set, without the need of external calibration. Simulations have allowed to verify the validity of the above described algorithm. The output generated by the CNN can be viewed in MRIcro medical image viewer (www.mricro.com), as shown in Fig. 7. Calculating the number of pixels corresponding to the injury

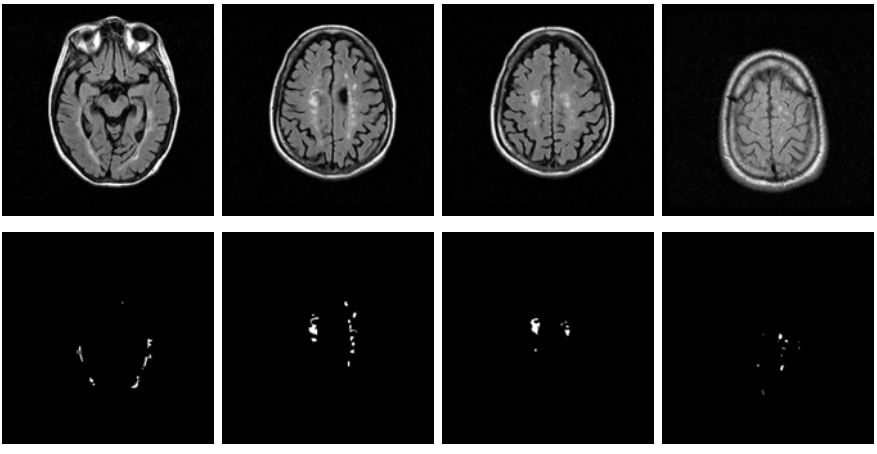


Fig. 7. Results of lesion-CNN segmentation on one of the MS patients

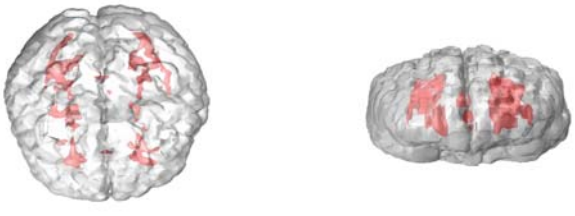


Fig. 8. A 3D reconstruction of lesion-CNN segmentation results. 3D reconstruction shows the typical spatial dissemination of lesions affecting white matter in MS.

and knowing the size of the voxel which scanning uses, it is possible to estimate the TLL for any patient. This method provides an important parameter to monitor the progress of the pathological disease. It should be emphasized that the results (Fig. 7) were obtained without changing the template from one slice to another. In fact, no manual thresholding is required during the segmentation process.

Obviously, operating with an ad hoc manual calibration of the network on any single slice, the algorithm is able to produce more precise results. Overlapping the slices and the output of CNN, it is possible to obtain a 3D reconstruction of the brain of the patient (Fig. 8), which displays the region of the brain tissue that presents multiple sclerosis.

5 Conclusions

In this paper we have presented an innovative CNN based method for automatically detect MS lesions. The results obtained by applying the proposed algorithm

have been very convincing, since CNN can determine most of the lesions in all the patients. The system could provide a useful MR support tool for the evaluation of lesions in MS, particularly to assess the evolution of the lesions. From a comparison with other existing methods in the literature on this topic, we can say that the results obtained with our method are effective and the threshold of recognition is currently at 70%. Furthermore, it should be emphasized the real improvement of the proposed method with respect to [5] for the greater accuracy of the system, its adaptation to different conditions of the stimuli, its ability to create 3D images of the injured areas of the brain, thus effectively supporting medical diagnosis.

References

1. Bilotta, E., Pantano, P.: Cellular Non-Linear Networks as a New Paradigm for Evolutionary Robotics. In: Iba, H. (ed.) Frontiers in Evolutionary Robotics, Vienna, Austria, pp. 87–108 (2008)
2. Chua, L.O., Roska, T.: Cellular Neural Networks and Visual Computing: Foundations and Applications. Cambridge University Press, Cambridge (2004)
3. Chua, L.O., Yang, L.: Cellular Neural Networks: Theory. IEEE Transaction on Circuits and Systems 35(10), 1257–1272 (1988)
4. Dice, L.R.: Measures of the amount of ecologic association between species. Ecology 26, 297–302 (1945)
5. Döhler, F., Mormann, F., Weber, B., Elger, C.E., Lehnertz, K.: A Cellular Neural Network based Method for Classification of Magnetic Resonance Images: Towards an Automated Detection of Hippocampal Sclerosis. In: Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 579–586 (2004)
6. Kek, L., Karacs, K., Roska, T.: Cellular Wave Computing Library (Templates, Algorithms and Programs), ver. 2.1. Cellular Sensory Wave Computers Laboratory, Hungarian Academy of Sciences (2007)
7. SPM8 - Statistical Parametric Mapping, http://www.fil.ion.ucl.ac.uk/spm/software/spm8/
8. Szabo, T., Barsi, P., Szolgay, P.: Application of Analogic CNN algorithms in Telemedical Neuroradiology. Journal of Neuroscience Methods 170(7), 2063–2090 (2005)
9. Valentino, P., Cerasa, A., Chiriaco, C., Nisticò, R., Pirritano, D., Gioia, M., Lanza, P., Canino, M., Del Giudice, F., Gallo, O., Condino, F., Torchia, G., Quattrone, A.: Cognitive deficits in multiple sclerosis patients with cerebellar symptoms. Multiple Sclerosis 15, 854–859 (2009)
10. Völk, K., Miller, J.F., Smith, S.L.: Multiple Network CGP for the Classification of Mammograms. In: Giacobini, M., Brabazon, A., Cagnoni, S., Caro, G.A.d., Ekart, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P., McCormack, J., O'Neill, M., Neri, F., Preuß, M., Rothlauf, F., Tarantino, E., Yang, S. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 405–413. Springer, Heidelberg (2009)

A Hybrid Evolutionary Algorithm for Bayesian Networks Learning: An Application to Classifier Combination

Claudio De Stefano, Francesco Fontanella, Cristina Marrocco, and Alessandra Scotto di Freca

Università di Cassino
Via G. Di Biasio, 43 02043 Cassino (FR) – Italy
{destefano,fontanella,cristina.marrocco,a.scotto}@unicas.it

Abstract. Classifier combination methods have shown their effectiveness in a number of applications. Nonetheless, using simultaneously multiple classifiers may result in some cases in a reduction of the overall performance, since the responses provided by some of the experts may generate consensus on a wrong decision even if other experts provided the correct one. To reduce these undesired effects, in a previous paper, we proposed a combining method based on the use of a Bayesian Network. The structure of the Bayesian Network was learned by using an Evolutionary Algorithm which uses a specifically devised data structure to encode Direct Acyclic Graphs. In this paper we presents a further improvement along this direction, in that we have developed a new hybrid evolutionary algorithm in which the exploration of the search space has been improved by using a measure of the statistical dependencies among the experts. Moreover, new genetic operators have been defined that allow a more effective exploitation of the solutions in the evolving population. The experimental results, obtained by using two standard databases, confirmed the effectiveness of the method.

1 Introduction

The idea of combining the results provided by different experts for improving the overall classification rate has been widely investigated in the literature and it is now an active area of research in the fields of Machine Learning and Pattern Recognition [8,9,6]. The rationale behind this idea is that the weakness of each single expert may be compensated without losing the strength of each of them, thus obtaining an overall performance that can be better than that of any single expert. Even if many studies have been published in the literature, which demonstrate, theoretically or empirically, the effectiveness of combining methods and their advantages over individual classifier models [11], their use may result in some cases in a reduction of the overall performance. This effect is mainly due to the fact that the responses provided by some of the experts may generate consensus on a wrong decision, even if other classifiers in the combining pool provided the correct class. Thus, the main problem to be solved is that of defining a combining rule able to solve these conflicts and to take the right classification decision even when the experts disagree.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 221–230, 2010.

We believe that an effective way to overcome the above drawbacks is that of considering the combined effects of the whole set of responses provided by the experts on the final decision, trying to estimate the statistical dependencies among them. In a previous work [4] we have exploited this idea by considering the above set of responses as representative of the collective behaviour of the combiner, and we have reformulated the classifier combination problem as a pattern recognition one, in which each sample to be recognized is represented by the set of class labels provided by the experts when classifying that sample. Thus, the role of the combiner is that of estimating the conditional probability of each class, given the set of labels provided by the experts for each sample of a training set. On the basis of these conditional probabilities, the final decision about the class to be assigned to an unknown sample is obtained by using the maximum a-posteriori probability (MAP) rule. In this way, the combining rule is automatically derived through the estimation of the conditional probability of each class. In our study, we adopted a Bayesian Network (BN) [13] to automatically infer the joint probability distributions between the outputs of the classifiers and the classes. This choice is motivated by the fact that BN's provide a natural and compact way to encode joint probability distributions through graphical models, and allow to gain understanding about complex problem domain. Even if the obtained results were very interesting, learning the structure of a BN, represented as Direct Acyclic Graph (DAG), is a NP-hard problem [1] and its exact solution becomes very soon computationally intractable as the number of random variables increases. This is the reason why standard algorithms search for suboptimal solutions by maximizing at each step a local scoring function which takes into account only the local topology of the DAG.

Moving from these considerations, we have proposed in [5] a new version of the combining method in which the structure of the BN is learned by means of an Evolutionary algorithm, using a direct encoding scheme of the BN structure. Such encoding scheme is based on aspecifically devised data structure, called *Multilist*, used for representing a DAG in each individual in the evolving population. The Multilist also allows an effective and easy implementation of the genetic operators. The experimental results confirmed the effectiveness of this approach showing some improvements with respect to the performance obtained by using our previous method. They also showed that the learning was quite slow, due to the complexity of the search space, and that the obtained solutions represented DAG structures with a large number of connections between nodes.

This paper represents a further development along this direction, in that we have developed a new hybrid evolutionary algorithm in which the exploration of the search space has been improved by using a measure of the statistical dependencies among the experts. Moreover, new genetic operators have been defined that allow a more effective exploitation of the solutions in the evolving population.

There are in the literature other few approaches for evolutionary learning of the Bayesian Network structure [12,15] but their main drawback is the use of data structures for representing DAG's in the form of adjacency matrix: this data structure makes difficult to implement genetic operators and does not guarantee that the new generated individuals are DAG's. The effect is twofold: on one hand, it is necessary to verify that new generated individuals satisfy the properties of DAG and this is a time consuming

task; on the other hand, the individuals not representing DAG's must be deleted making less efficient the exploration of the search space.

The remainder of the paper is organized as follows: Section 2 illustrates the architecture of the combining method. Section 3 discusses the evolutionary algorithm for evolving DAG's. Section 4 reports the experimental results, while Section 5 reports some concluding remarks.

2 The Architecture of the Combiner

Consider the responses $e_1, \ldots, e_L$ provided by a set of L classifiers (experts) for an input sample x in a N class problem, and assume that such responses constitute the input to the combiner, as shown in figure 1. The combiner can be defined as a "higher level" classifier that works on a L-dimensional discrete-values feature space.

It is assumed that the combiner uses a supervised learning strategy, where the learning procedure consists in the observation of the set of responses $e = \{e_1, \ldots, e_L\}$ and of the "true" class label u of a sample x, for computing $p(u|e_1, \ldots, e_L)$. Once this conditional probability has been learned, the combiner provides the output $\widehat{u}$ for each unknown input sample, as the most probable class given the expert observations, by the following expression:

$$\widehat{u} = \arg\max_{u \in C} p\,(u|e_1, ..., e_L) \tag{1}$$

where C is the set of classes. Considering the definition of conditional probability and omitting the terms not depending on the variable u to be maximized, Eq. (1) can be rewritten as:

$$\widehat{u} = \arg\max_{u \in C} p\,(u, e_1, ..., e_L). \tag{2}$$

that involves only the joint probabilities $p\,(u, e_1, ..., e_L)$. Hence the combining problem represented in Eq. (1) is equivalent to that of maximizing the joint probability in Eq. (2): this problem may be effectively been solved by using Bayesian Networks.

In the next subsections we will introduce some basic concepts and some mathematical properties of Bayesian Networks, as well as the basic concepts relative to Bayesian Network learning. A more detailed description of the Bayesian Networks theory can be found in [7].

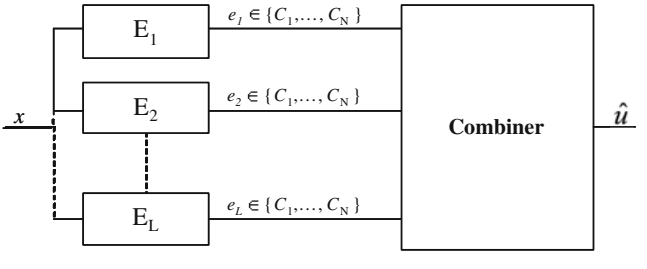


Fig. 1. The architecture of our combiner

2.1 Bayesian Networks Properties

A BN allows the representation of a joint probability law through the structure of a Direct Acyclic Graph (DAG). The nodes of the graph are the variables, while the arcs are their statistical dependencies. An arrow from the generic node i to node j has the meaning that j is conditionally dependent on i, and we can refer to i as the parent of j. For each node, a conditional probability quantifies the effect that the parents have on that node.

Considering that a DAG describes the statistical dependencies among variables, the conditional probability distribution of a random variable e_i, given all the other, can be simplified as follows:

$$p(\, e_i \,|\, pa_{e_i}, nd_{e_i}) = p(\, e_i \,|\, pa_{e_i}\,) \tag{3}$$

where pa_{e_i} indicates the set of nodes which are parents of node e_i, and nd_{e_i} indicates all the remaining nodes. Eq (3), known as causal Markov property, allows the description of the joint probability of a set of variables $\{u, e_1, \ldots, e_L\}$ as:

$$p\,(u, e_1, \ldots, e_L) = p\,(\, u \,|\, pa_u\,) \prod_{e_i \in L} p\,(\, e_i \,|\, pa_{e_i}\,) \tag{4}$$

In case of a node having no parents, the conditional probability coincides with the a priori probability of that node. It is worth noticing that the node u may be parent of one or more nodes of the DAG. Therefore, it may be useful to divide the L nodes of the DAG in two groups: the first one L_u contains the nodes having the node u among their parents, and the second one $L_{\overline{u}}$ the remaining nodes. With this assumption, the Eq. (4) can be rewritten as:

$$p(u, e_1, \ldots, e_L) = p(u|\, pa_u) \prod_{e_i \in L_u} p(e_i|\, pa_{e_i}) \prod_{e_i \in L_{\overline{u}}} p(e_i|\, pa_{e_i}) \tag{5}$$

It is worth noticing that the last term of Eq. (5) is constant in the variable u and then it can be discarded while maximizing with respect to u. Therefore the Eq. (2) becomes:

$$\widehat{u} = \arg\max_{u \in C} p(u|\, pa_u) \prod_{e_i \in L_u} p(e_i|\, pa_{e_i}) \tag{6}$$

In such a way the approach detects the experts that do not add information to the choice of $\widehat{u}$, or, in other words, selects a reduced set of relevant experts whose outputs are actually used by the combiner to provide the final output.

2.2 Learning Bayesian Network

BN estimates the joint probability distribution of random variables by a supervised procedure that allows to learn, from a training set of examples, both the network structure, which determines the statistical dependencies among variables, and the parameters of such a probability distribution. Let us denote with S^h the structure of the DAG and with D a training set of samples. In our study, each sample of D, corresponding to a pattern x to be classified, is made of both the ordered list of labels provided by the classifiers

for that pattern, and the "true" label of x. Under the assumption made in [7], learning structure and parameters from data means maximizing the function $p(D|S^h)$.

According to the chain rule property of random variables in a DAG, the likelihood $p(D|S^h)$ can be factorized as follows:

$$p(D|S^h) = Score(S^h) = \prod_{i=0}^{L} localscore(i) \tag{7}$$

where $localscore(i)$ is a function formally defined in [7]. It is worth noticing that any change in S^h requires that only the local scores of the nodes affected by that change need to be updated for computing $Score(S^h)$.

The function $localscore(i)$ measures how much the expert e_i is statistically dependent on the set of its parent nodes pa_{e_i} and it is computed as follows:

$$localscore(i) = \prod_{m=1}^{S_{pa_{e_i}}} \frac{\Gamma(\alpha_{im})}{\Gamma(\alpha_{im} + N_{im})} \prod_{k=1}^{N} \frac{\Gamma(\alpha_{imk} + N_{imk})}{\Gamma(\alpha_{imk})}. \tag{8}$$

where $\Gamma(.)$ is the Euler Gamma function and $S_{pa_{e_i}}$ is the total number of states of pa_{e_i}. Considering that in our case e_i has N states corresponding to each of the classes in C, if the expert e_i has q parents $S_{pa_{e_i}} = N^q$. This is the reason why for each response provided by the expert e_i, a vector of q terms representing the answers of the parent nodes of e_i must be analyzed. The term N_{imk} represents how many times pa_{e_i} in the state m and the expert e_i is in the state k. The term N_{im}, instead, represents how many times pa_{e_i} is in the state m independently from the response provided by the expert e_i. The terms α_{im} and α_{imk} are normalization factors.

Summarizing, the learning of a Bayesian Network can be performed by finding the DAG structure S^h, which maximizes the function $Score(S^h)$. To solve this problem, we have defined an evolutionary algorithm which encodes a DAG structure in each individual and uses the function $Score(S^h)$ as fitness function. In the next Section, a detailed description of the proposed evolutionary algorithm will be provided.

3 Evolutionary Bayesian Network Learning

The algorithm that we have implemented for Bayesian network learning encodes DAG's through a specifically devised data structure called *multilist*. The algorithm consists of two phases: a preliminary phase and a search phase. In the preliminary phase, for each couple of variables, a measure of their statistical dependencies is computed by means of the mutual information [3] and stored in a matrix M. In the search phase, a generational evolutionary algorithm is used to find the Bayesian network for the combining problem. Before presenting the details of our hybrid algorithm, let us describe the encoding scheme and the genetic operators.

3.1 DAG Encoding and Genetic Operators

In the DAG terminology a *source* is a node with no incoming arcs while a *sink* is a node with no outgoing arcs. In a DAG nodes are partially ordered: if it exists a directed path

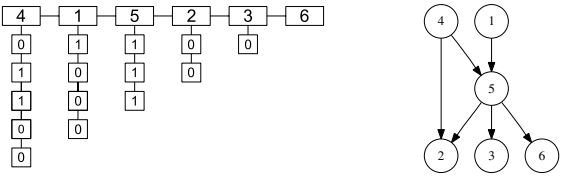


Fig. 2. A multilist (right) and the encoded DAG's(left)

from node i to node j then i precedes j, otherwise it is not possible to define an ordering between them. The data structure that we have devised for encoding DAG, called *multilist* (ML), consists of two basic lists. The first one, called *main list*, contains all the nodes of the DAG ordered according to the partial ordering previously defined. This implies that *source* nodes occupy the first positions, while *sink* node, the last positions. Moreover, nodes having both incoming and outgoing arcs are inserted in the *main list* after their parents. To each node of the *main list* is associated a second list called *sublist*, representing the outgoing connections among that node and the other nodes in the DAG. More specifically, if s_i is the *sublist* associated to the $i-th$ element of the *main list*, then it contains information about the outgoing arcs possibly connecting the $i-th$ element and the other elements following it in the *main list*, ordered according to the position of such elements. Since an arc may be present or not, each element of a *sublist* contains a binary information: 1 if the arc exists, 0 otherwise (see figure 2). Note that the length of the *sublists* decreases as the position of the element in the main list increases: assuming that there are K nodes in the DAG, the first sublist contains $(K-1)$ elements, the second one $(K-2)$ elements and so on. In fact, the informations about the arcs connecting a node and the previous ones in the main list are already expressed in the previous sublists. As a consequence, the sublist of the last element in the *main list* is void. Thus a ML has a triangular shape: the base of the triangle is the main list and contains K elements, while the height is represented by the first sublist containing $(K-1)$ elements.

As regards the genetic operators, we have defined two mutation operators which can modify a ML in two different ways: (i) swapping two elements of the main list; (ii) adding and/or deleting one or more arcs in a sub list. In the following these mutations will be called respectively m and s mutation. The m–mutation performs a permutation on the elements of the main list, but leaves unchanged the connection topology of the ML. This mutation consists of two steps: (i) randomly pick two elements in the main list and swap their position; (ii) modify sublist elements in such a way to restore the connection topology as it was before the step (i). It is worth noticing that the m–mutation generates a new ordering of the variables, which modifies the directions of the existing arcs in the DAG, but preserves dependencies between variables. If we consider the DAG in figure 2, for instance, the swap between the second and the fourth node in the main list changes only the directions of the arcs connecting the couples of nodes $(1,5)$ and $(5,2)$. Finally, given a multilist, this operator is applied to its main list according to a probability value p_m. The s–mutation, instead, modifies the values of the *sublist* elements. For each element of the sublists, p_s represents the probability of changing

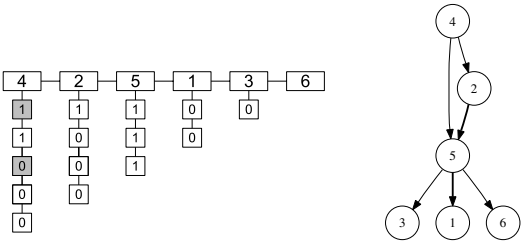


Fig. 3. (a) The multilist generated by swapping the second and the fourth node of the main list of the DAG in figure 2(b); the sublist elements modified to restore the connection topology are shaded. (b) The encoded DAG; the inverted arcs are in bold.

its value from 0 to 1, or viceversa. Thus the effect of this operator is that of adding or deleting arcs in the DAG. Such an operation is applied with probability p_s.

3.2 The Hybrid Evolutionary Algorithm

In order to implement the hybrid approach, we have computed in the preliminary phase the mutual information $I(i, j)$ between each couple of nodes in the DAG, and we have stored such values in a matrix M. We have also stored the values m_{min} and m_{max}, representing the minimum and maximum value in M, respectively. The rationale behind this idea is that of using the mutual information between each couple of variables to decide whether or not the an arc connecting the corresponding nodes in the DAG may be added during the search process. In particular, for each couple of variables i and j, if $M[i, j]$ is less than a given threshold θ, the arcs $i \longrightarrow j$ and $j \longrightarrow i$ are excluded from the search space, meaning that they are not considered neither during the random generation of the initial population, nor during the evolution by the s-mutation operator. It is worth noting that the choice of the value for the threshold θ is crucial: if a too high value is chosen, most of the arcs are pruned away while, using a too low value, all the arcs are considered and the search space is not reduced at all. To cope with this problem we decided to evolve the value of θ, putting this information in the genotype. As a consequence, each individual considers a different search space of DAG's: the value of θ is randomly initialized at the beginning of the evolution and is dynamically modified during the subsequent generations. As the evolutionary search proceeds, individuals having improper values of θ will eventually be eliminated.

The evolutionary algorithm starts by generating an initial population of P individuals. The initialization of each individual consists of three steps:

1. a value is randomly chosen in the range $[m_{min}, m_{max}]$ and assigned to θ;
2. the stochastic variables, representing the responses of the experts to be combined, are randomly associated to the nodes in the main list;
3. arcs are initialized taking into account both the value of θ previously chosen, and the probability p_a of inserting a connection between two nodes. In practice, for each couple of nodes i and j, where i precedes j in the main list, the corresponding arc

$i \longrightarrow j$ is added if and only if the value $M[i,j] > \theta$ and the the function *flip*(p_a)[1] returns TRUE.

The evolution process is repeated for n_g generations. At each generation the following steps are executed:

1. the fitness of the individuals in the current population is evaluated using the likelihood $p(D|S^h)$ defined in eq. 7;
2. the best e individuals are selected and copied in the new population in order to implement an elitist strategy;
3. $(P - e)$ individuals are selected: in order to control loss of diversity and selection intensity, we have used the tournament selection method;
4. for each selected individual, m–mutation and s–mutation are applied with probability p_m and p_s, respectively, and a new value for θ is obtained adding or subtracting to the previous one an offset in the range $[0, \Delta_\theta]$;
5. the modified individuals are then added to the new population.

4 Experimental Results and Discussion

The proposed method has been tested on two standard databases, namely the Multiple Feature (MFeat) and the IMAGE database from the UCI Machine Learning Repository. The first database contains handwritten digits, while the second one images with different textures. In each experiment we split the samples of each class in three statistically independent sets: TR1, TR2 and TS. TR1 is used for training each single classifier, while TR2 and TS are used to collect the responses of each single classifier on their samples. The responses collected on TR2 are used for training the combiner, while those collected on TS are used for evaluating the combiner performance. As with respect to the classifiers, we have used two different schemes: a Back-Propagation neural network (BP) [14] and a Learning Vector Quantization neural network (LVQ) [10]. During a training phase, each classifier was separately trained on TR1. For each database, the pool of experts has been obtained by generating an ensemble of BP nets and an ensemble of LVQ nets.

In the first experiment, the 2000 available samples have been divided in the following way: TR1 contains 700 samples, while both TR2 and TS include 650 samples. The pool of experts has been obtained by combining each classification scheme with the six feature sets included in the MF Database, totaling twelve experts. In the second experiment, TR1 was made of 210 samples, 30 per each of the 7 classes, while both TR2 and TS contain 1050 elements, 150 per class. The pool of experts has been obtained by combining two ensembles of BP and LVQ nets, each containing 10 different experts obtained by randomly initializing the nets.

As regards the values of the evolutionary parameters, they have been determined by a set of preliminary experiments and are summarized in Table 1. The probability p_s to apply the s–mutation is equal to $1/N_s$, where N_s is total number of elements in the

[1] The function flip(x) returns the value 1 with a probability x and the value 0 with a probability $(1 - x)$.

Table 1. Values of the basic evolutionary parameters used during the experiments

Parameter	symbol	value
population size	P	100
tournament size	$\mathcal{T}$	6
elitism size	e	2
m–mutation probability	p_m	0.8
s–mutation probability	p_s	$1/N_s$
arc probability	p_a	0.1
offset range	Δ_θ	0.01
number of generations	N_g	2000

Table 2. Comparison of Classification Results

	Best Expert	Majority vote	BN Combiner	Evo–BN1 Combiner	Evo–BN2 Combiner
MFeat	96.89%	97.33%	99.10%	99.28%	100%
IMAGE	91.00%	89.14%	91.90%	92.30%	93.42%

sublists of an individual. It is worth noting that this value depends on the number K of nodes in the DAG to be learned, since N_s is equal to $K(K-1)/2$. Thus this probability value is such that, on the average, only one sublist element is modified when it is applied. The results of our EC-based method for Bayesian networks learning (EVO-BN2 in the following) have been compared with those obtained by our previous EC-based approach presented in [5] (EVO-BN1 in the following). EVO–BN1 also uses the multilist data structure for DAG encoding, but a different set of genetic operators. Table 2 shows the classification results: the first column reports the results of the best single expert in the pool, while the second column reports the results of the Majority Vote Combining rule [11]. The third column reports the results of the Bayesian Combiner (BN) implemented by using a standard greedy search algorithm [2] to learn the DAG structure. The fourth and the fifth columns, respectively, show the results of the Evolutionary Bayesian Combiners Evo–BN1 and Evo–BN2. Note that we have reported only the best results obtained in each experiment: the average values, in fact, are practically identical because the standard deviations are very small, exhibiting values always lower than 10^{-3}. The data reported in the Table 2 show that Evo–BN2 improves the performance with respect to Evo–BN1 on both datasets. As regards the MFeat dataset EVO–BN2 has further improved the good rates obtained by EVO-BN1, reducing to zero the error rate on the test set. Also on the IMAGE dataset, the results are better than those obtained with EVO–BN1. These results confirm that the use of a hybrid strategy, together with the new definition of the genetic operators, allows us to improve the obtainable results.

5 Conclusions

A new EC-based algorithm for Bayesian Network learning has been presented. The proposed approach uses a special data structure called multilist specifically devised for

encoding DAG's. The use of the multilist makes the Bayesian Networks learning more efficient, because it intrinsically encodes DAG's, avoiding the time consuming task of checking the acyclicity property. Moreover, in order to improve the effectiveness of the proposed method with respect to our previous implementations, two strategies have been adopted: (i) the operators has been modified in order to better preserve the structure of the multilist to which they are applied; (ii) an hybrid approach which exploits the knowledge given by the measure of the mutual information has been used in order to reduce the search space.

References

1. Chickering, D.M., Geiger, D., Heckerman, D.: Learning bayesian networks is np-hard. Tech. rep. (1994)
2. Cooper, G.F., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning 9(4), 309–347 (1992)
3. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, Hoboken (2006)
4. De Stefano, C., D'Elia, C., Marcelli, A., Scotto di Freca, A.: Classifier combination by bayesian networks for handwriting recognition. International Journal of Pattern Recognition and Artificial Intelligence 23(5), 887–905 (2009)
5. De Stefano, C., Fontanella, F., Marcelli, A., Scotto di Freca, A.: Learning bayesian networks by evolution for classifier combination. In: ICDAR 2009: Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, pp. 966–970. IEEE Computer Society, Los Alamitos (2009)
6. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
7. Heckerman, D.: A tutorial on learning with bayesian networks. Tech. rep., Learning in Graphical Models (1995)
8. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Trans. Pattern Anal. Mach. Intell. 16(1), 66–75 (1994)
9. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
10. Kohonen, T.: Self organizing map. Springer, Berlin (1995)
11. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
12. Larranaga, P., Poza, M., Yurramendi, Y., Murga, R.H., Kuijpers, C.M.: Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(9), 912–926 (1996)
13. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
14. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
15. Wong, M.L., Leung, K.S.: An efficient data mining method for learning bayesian networks using an evolutionary algorithm-based hybrid approach. IEEE Trans. Evolutionary Computation 8(4), 378–404 (2004)

Towards Automated Learning of Object Detectors

Marc Ebner

Eberhard-Karls-Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Abt. Rechnerarchitektur, Sand 1, 72076 Tübingen, Germany
marc.ebner@wsii.uni-tuebingen.de
http://www.ra.cs.uni-tuebingen.de/mitarb/ebner/welcome.html

Abstract. Recognizing arbitrary objects in images or video sequences is a difficult task for a computer vision system. We work towards automated learning of object detectors from video sequences (without user interaction). Our system uses object motion as an important cue to detect independently moving objects in the input sequence. The largest object is always taken as the teaching input, i.e. the object to be extracted. We use Cartesian Genetic Programming to evolve image processing routines which deliver the maximum output at the same position where the detected object is located. The graphics processor (GPU) is used to speed up the image processing. Our system is a step towards automated learning of object detectors.

1 Motivation

A human observer has no problems in identifying different objects in an image. How do humans learn to recognize different objects in an image? Enabling a computer vision system to perform this feat is a daunting task. However, we try to work towards this goal. An ideal computer vision system would be able to automatically learn different object detectors from scratch. It is obviously highly desirable to develop self-adapting and self-learning vision systems which work without human intervention [4]. We have developed an evolutionary computer vision system which is able to automatically generate object detectors without human intervention.

Our system is based on a previously developed evolutionary vision system using GPU accelerated image processing [6]. Input to the system is a continuous stream of images. Each input image is processed by several different computer vision algorithms. The best algorithm is used to supply the overall output, i.e. to detect objects in the input image. Evolutionary operators are used to generate new alternative algorithms. The original system required user interaction to identify the objects to be detected. We have extended this system such that no user interaction is required.

For humans, motion serves as an important cue to identify interesting objects. Our system detects differences between consecutive images in order to detect independently moving objects in the image. Each detected object is equipped with

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 231–240, 2010.

a 2D motion model which describes the motion of the object on the screen [3]. This motion model is continuously updated based on the motion differences between two consecutive images. By directly transforming a sequence of difference images into a 2D motion model, the computational resources needed to compute the teaching input, is reduced to a minimum. As soon as one or more objects have been detected in the input sequence, the system always focuses on the largest object. The center of the object is taken as the teaching input.

The paper is structured as follows. In Section 2 we give a brief overview about related research in evolutionary computer vision. Section 3 describes how motion is used to obtain the teaching input. The GPU accelerated evolutionary vision system is described in Section 4. Experiments are presented in Section 5. Conclusions are provided in Section 6.

2 Evolutionary Computer Vision

Evolutionary algorithms can be used to search for a computer vision algorithm when it is not at all clear what such an algorithm should look like. Evolutionary algorithms can also be used to improve upon an existing solution. Work in evolutionary computer vision started in the early 1990s. Lohmann has shown how an Evolution Strategy may be used to find an algorithm which computes the Euler number of an image [15]. Early research focused on evolving low-level operators, e.g. edge detectors [8] or feature detectors [20]. However, evolutionary algorithms were also used for target recognition [12].

In theory, evolutionary methods can be used to evolve adaptive operators which would be optimal or near optimal for a given task [7]. Poli noted very early on, that Genetic Programming [13] would be particularly useful for image processing [19]. Genetic Programming has been used to address a variety of different tasks in computer vision. Johnson et al. have evolved visual routines using Genetic Programming [11]. Current work in evolutionary computer vision ranges from the evolution of low-level detectors [22], to object recognition [14] or even camera calibration [9]. Cagnoni [2] gives a taxonomic tutorial on evolutionary computer vision.

Experiments in evolutionary computer vision usually require enormous computational resources. Each individual of the population has to be evaluated over several generations. That's why experiments in evolutionary computer vision are usually performed off-line. A notable exception (also working with on-line learning) is the work of Mussi and Cagnoni [17]. In our context, multiple alternative image processing algorithms have to be applied to each incoming image. This is only possible through the use of GPU accelerated image processing. Before we describe our evolutionary computer vision system, we first describe how the teaching input is obtained from the input sequence.

3 Fast Detection of Moving Objects in Image Sequences

We have developed a fast method for detecting independently moving objects in image sequences. It is assumed that the camera remains stationary while the

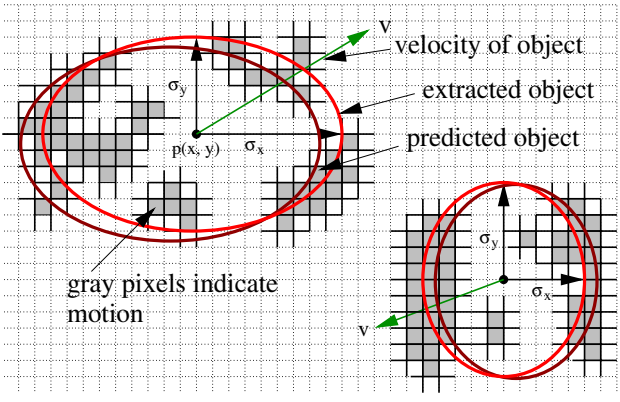


Fig. 1. Differences between two consecutive images are assigned to the nearest object which is predicted from a previous time step. The coordinates of the object predictions (x, y) for the current time step are given by the center of gravity which is computed using all pixels assigned to the object. Two extracted objects are shown.

image sequence is taken. If the camera itself moves, then information about the ego-motion of the camera could be used to compute a quasi-stationary camera sequence [3].

The method is fast, because image processing operations are reduced to a minimum. Image processing operations are costly because they are applied at least once to every pixel. Operations such as convolution or averaging are particularly costly if they are applied in image space because they require a sliding window and multiple surrounding pixels are accessed for each image pixel.

Thus, we only compute a difference image between two successive images and use this information to update object hypotheses [3]. The approach could be considered to be a minimalistic variant of a particle filtering approach [1,10] where only a single particle is used per object. We compute the average of the differences of the three channels red, green and blue. Differences smaller than than 10% from the maximum (assumed to be noise) are set to zero. We continue by only considering differences larger than this threshold.

Let us assume that we have a set of previously extracted objects, i.e. a prediction where objects will be located in the current image. Each object consists of a center of gravity with coordinates $\mathbf{p} = (x, y)$ and also has an associated velocity $\mathbf{v} = (v_x, v_y)$ with which it moves across the image (Figure 1). Each object also has an associated standard deviation in x- and y-direction (σ_x, σ_y). The standard deviations determine the extent of the object. Each pixel with a difference larger than the threshold contributes to the nearest object. In case a pixel difference cannot be assigned to any object prediction, a new object is created. For newly created object predictions, the center of gravity as well as the standard deviations describing the shape of the object are continuously updated as new pixels are assigned to it.

The center of gravity for the updated object position is computed using all pixels which are located within a distance no more than twice the standard deviation from the center of the object. Pixels further away are assumed to belong to a different object. The standard deviations describing the extent of the object are updated using the same pixels. Let $\mathbf{p}(t_0)$ and $\mathbf{p}(t_1)$ be the positions of the object at time steps t_0 and t_1 respectively. The difference $\mathbf{d} = p(t_1) - p(t_0)$ between the object position for the previous image and the current image is used to update the motion vector of the object. We filter this difference to obtain a smooth approximation of the actual motion vector $\mathbf{v}$ using

$$\mathbf{v}(t_1) = 0.9\mathbf{v}(t_0) + 0.1\mathbf{d}. \tag{1}$$

If an object does not have any associated pixel differences, then the old motion vector is simply added to the center of gravity to predict the new object position for the next time step. If an object does not have any associated pixel differences for three consecutive images, then the object is deleted.

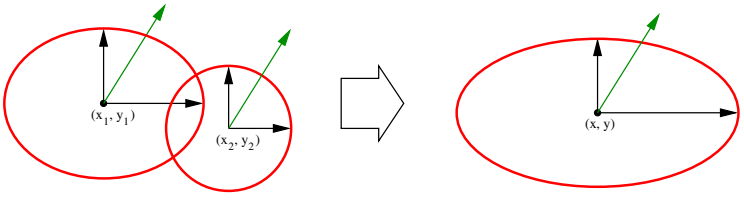


Fig. 2. Nearby objects are merged if they are close to each other

After existing objects have been updated and new objects have been detected, we iterate over all objects to find objects which need to be merged (Figure 2). Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the center of gravities of two different objects at the same time step. New objects (for which no motion vector exists yet) are merged if the distance between their center of gravities is smaller than twice the sum of their standard deviations, i.e. if

$$\mathbf{p}_1 - \mathbf{p}_2 \leq 2(\sigma_1 + \sigma_2) \tag{2}$$

with $\sigma_i = \sqrt{\sigma_{x,i}^2 + \sigma_{y,i}^2}$. Existing objects are merged only if the distance between their center of gravities is less than the sum of their standard deviations. They are also merged if the distance is less than twice the sum of their standard deviations provided that they approximately move in the same direction, i.e. their motion vector differs by less than 10%.

Figure 3 shows how a moving objects are detected and tracked over several frames in two image sequences. We will use the same sequences to evolve a detector for these objects.

Fig. 3. Moving object detected in two video sequences. The yellow circle marks the detected object. (a) radio-controlled car (b) toy train.

4 A GPU Accelerated Evolutionary Vision System

Ebner [6] has developed a GPU accelerated evolutionary vision system. When searching for a solution to a computer vision algorithm, one basically has to assemble computer vision operators in the correct order and also has to decide with which parameters these operators are applied. We are using Cartesian Genetic Programming [16] to automatically search the space of optimal algorithms.

The system works with a $(n_x \times n_y)$ matrix of image processing operators as shown in Figure 4. In addition to this matrix, a set of n_1 high level image processing operators are applied to the input image. We will refer to all of these operators as processing nodes. Each individual of the population codes for an arrangement of image processing operators. High level operators use the original image as input and also create an image as output. Low level operators can have either one or two inputs. Low level operators only perform point operations, i.e. low level operators can be computed by iterating once over all image pixels.

High level operators include operators such as the original image at different scale levels or with a small offset, derivatives in the x- and y-direction, the Laplacian, the gradient magnitude, computation of gray scale images from RGB, segmentation or a convolution. Note that the individuals only have access to a single image frame. We deliberately do not supply two different frames to the system. Individuals should be able to recognize objects in single image frames.

The output of the high level operators is processed by the low level operators inside the $(n_x \times n_y)$ matrix. Data is always fed from left to right. A particular node has only access to the data stored in the previous column. The wiring for this matrix is completely under the control of evolution. Low level operators include arithmetic operations, step functions, gate functions, maximum and minimum operations and related functions which access the output from either one or two nodes. The parameters used inside the nodes are also under the control

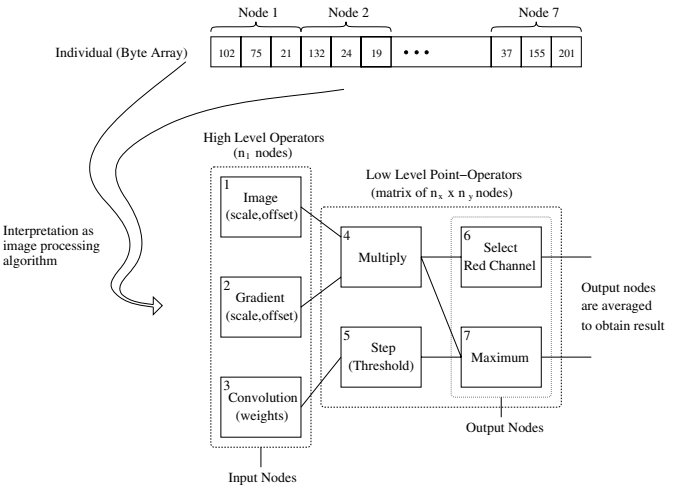


Fig. 4. Sample individual. A linear byte array is mapped to an image processing program consisting of n_1 high level operators and a processing matrix of $n_x \times n_y$ low level, point operators. The output is averaged to obtain the resulting image.

of evolution. A full description of the operators is given by Ebner [6]. The same system is used here except that the constants (0, 0.5 and 1) have been moved from the set of high level operators to the set of low level operators.

Most of the operators are taken directly from the specification of the OpenGL Shading Language (OpenGLSL) [21]. The OpenGL Shading Language was used to accelerate the image processing because it allows easy access to scale spaces which is particularly important for the implementation of the high level operators. It is not clear whether a CUDA [18] implementation would provide any advantage to the GPU acceleration method used here.

Each individual transforms the input image into some other image, the output image. The output image is computed by averaging the output of the n_y rightmost nodes. In order to determine where the object is detected by an individual, we iterate over all image pixels. The position with the maximum output (RGB components are treated as an integer) is taken as the object position. If more than one pixel has the same maximum value, we compute the center of gravity of these pixels. More than 20 pixels having the same maximum value are discouraged by assigning a bad fitness value. We want to the system to clearly mark the detected object in the image.

This representation is referred to as a $n_1 + n_x \times n_y$ representation. It is fully described by Ebner [5,6]. The system works with a linear genotype. Each node has three associated parameters. The first parameter determines the operator used. The remaining two parameters are either used as parameters for the image processing operator or are used to determine from which previous node the input is received. Each parameter is represented by 8 bits in the genotype.

Each genotype is mapped to the representation shown in Figure 4. A set of n_p individual constitutes the parent population. From the parent population, n_o offspring are generated by applying genetic operators. An additional n_r offspring are generated randomly. Mutation and crossover are used as genetic operators. For each incoming image, parent as well as offspring are evaluated. The best n_p individuals among parents and offspring become the parents for the next input image. When selecting new parents, no double fitness values are allowed. Individuals with the same fitness are assumed to be identical. Using this approach we try to encourage a diverse population of parent individuals.

The fitness of an individual is simply the Euclidian distance between the position detected by the individual and the desired position which is computed using the method described in the previous section.

5 Experiments

For our experiments, we have used $n_p = 3$ parents which generate $n_o = 20$ offspring and $n_r = 20$ randomly generated offspring. Offspring are generated using two point crossover with a crossover probability of $p_{\text{cross}} = 0.5$. The remaining individuals are generated through mutation. The mutation operator either uses a GA-style mutation with a bit wise probability of $p_{\text{mut}} = \frac{2}{l}$ where l is the length of the genotype in bits, or increases or decreases one of the parameters by one. This is to allow also smooth changes of the parameters. A gray code could have been used instead to achieve the same effect.

Given the automated method to extract moving objects, we have objective data to work with and we can rigorously analyze how the method works. This is in contrast to the previous method, where the user had to manually select the object which should be extracted. We work with two video sequences (sample images are shown in Figure 3). The first video sequence shows a radio-controlled car moving around. It consists of 2097 image frames (1m:24s) of size 512×288. The car has a pronounced color which only occurs on the car and not on other objects shown in the sequence. In other words, it is quite easy to come up with an object detector for this car. A simple color filter will do the job. The second video sequence shows a toy train moving around on a track in circles. It consists of 1581 image frames (1m:03s) of size 512×288. The toy train is mostly colored in yellow and red. The same red color can also be found on a wagon which is always present in the image. The yellow color is also shown on the wagon. However, the yellow color on the wagon takes up only a smaller area compared to the yellow on the train.

For our experiments, we turn the evolutionary process on, as long as the desired position differs by more than 25 pixels from the detected position by the individual. The size of the car is approximately 45×50 pixels and the toy train is approximately 34×24 pixels. Evolution is turned off if the detected position is very close to the actual position, i.e. the difference between the two is less than 10 pixels for five consecutive frames. If this happens, then only the parent individuals are evaluated and no new offspring are generated. Once the

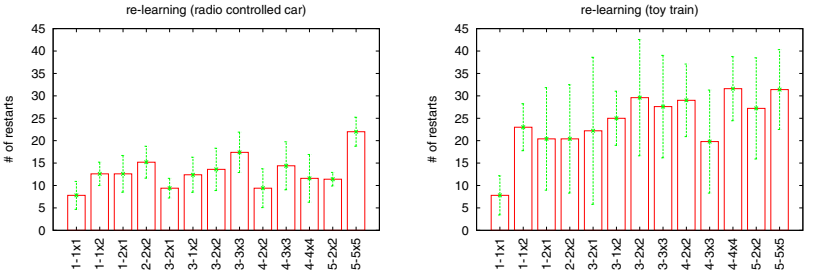


Fig. 5. Average number of restarts for the two image sequences (radio-controlled car and toy train). The standard deviation is also shown.

fitness, i.e. the error, rises to more than 25 pixels, then the evolutionary process is turned on again. Note that the individuals do not have to be re-initialized due to the continuous injection of random individuals into the population.

We evaluate how easy or difficult it is to evolve solutions using different $n_1 + n_x \times n_y$ representations. For both image sequences, we measure how often evolution has to be restarted. As described above, evolution has to be restarted if the object is no longer tracked. If evolution has to be restarted only once in a while, then the evolved detectors are very general. If evolution has to be restarted frequently, then the detectors are not general. Such detectors depend on the orientation and/or the size of the object in the image. Figure 5 shows the results for both image sequences. Five experiments were carried out using different random seeds to compute the average.

For the radio controlled car, evolution was only required for 4.5% of the image frames (averaged across all representations and experiments). For 95.5% of the image frames, the object was successfully detected. The object was detected on average with an accuracy of 7 pixels. For the toy train, evolution was only required for 14.6% of the image frames (averaged across all representations and experiments). For 85.4% of the image frames, the object was successfully detected. The object was detected on average with an accuracy of 8 pixels.

It is apparent that the toy train is more difficult to recognize. The data also shows that the problem gets more difficult as the size of the representation is increased. Thus, we want to keep the complexity of the representation minimal while still making sure that the solution is still inside the search space.

Figure 6 shows how long evolution was required to come up with a solution depending on the representation used. Again, the more complex the representation, the longer it took to find good solutions. The toy train sequence is clearly more difficult for the system. For the toy train sequence, it is not sufficient to only use a color detector. The system also has to take the arrangements of the colors into account. To come up with an effective detector for the toy train, the system would have to archive good solutions and to create an overall detector which would recombine the output of several archived detectors. Establishing an archive of detectors will be our next research goal.

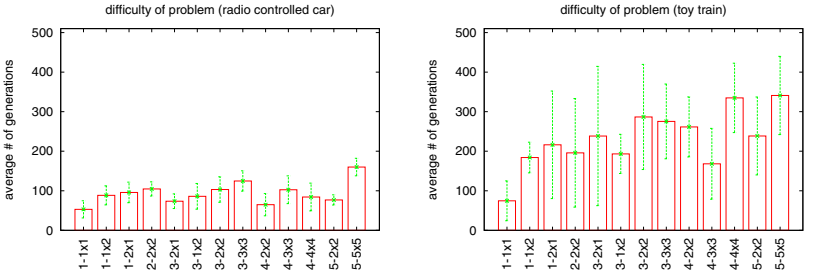


Fig. 6. Number of evolutionary steps. The standard deviation is also shown.

6 Conclusions

We have created a GPU accelerated evolutionary image processing system. The system automatically detects moving objects in a video sequence taken with a stationary camera. The coordinates of the detected objects are used to evolve object detectors which are able to recognize the object in a single image. We deliberately use one cue (motion) to train an object detector which is able to recognize objects in images when this cue is not available. The long term goal of this research is to come up with a system which automatically generates object detectors. Our system is a step towards automated learning of object detectors.

References

1. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking. IEEE Trans. on Signal Processing 50(2), 174–188 (2002)
2. Cagnoni, S.: Evolutionary computer vision: a taxonomic tutorial. In: 8th Int. Conf. on Hybrid Int. Systems, pp. 1–6. IEEE Computer Society, Los Alamitos (2008)
3. Ebner, M.: Extraction of moving objects with a moving mobile robot. In: Salichs, M.A., Halme, A. (eds.) 3rd IFAC Symposium on Intelligent Autonomous Vehicles, Madrid, Spain, vol. II, pp. 749–754. Elsevier Science, Amsterdam (1998)
4. Ebner, M.: An adaptive on-line evolutionary visual system. In: Hart, E., Paechter, B., Willies, J. (eds.) Workshop on Pervasive Adaptation, Venice, Italy, pp. 84–89. IEEE, Los Alamitos (2008)
5. Ebner, M.: Engineering of computer vision algorithms using evolutionary algorithms. In: Blanc-Talon, J., Philips, W., Popescu, D., Scheunders, P. (eds.) Advanced Concepts for Intelligent Vision Systems, Bordeaux, France, pp. 367–378. Springer, Berlin (2009)
6. Ebner, M.: A real-time evolutionary object recognition system. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 268–279. Springer, Heidelberg (2009)
7. Ebner, M., Zell, A.: Evolving a task specific image operator. In: Poli, R., Voigt, H.-M., Cagnoni, S., Corne, D.W., Smith, G.D., Fogarty, T.C. (eds.) EvoIASP 1999 and EuroEcTel 1999. LNCS, vol. 1596, pp. 74–89. Springer, Heidelberg (1999)

8. Harris, C., Buxton, B.: Evolving edge detectors with genetic programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming, Proceedings of the 1st Annual Conference, Stanford University, pp. 309–314. The MIT Press, Cambridge (1996)
9. Heinemann, P., Streichert, F., Sehnke, F., Zell, A.: Automatic calibration of camera to world mapping in RoboCup using evolutionary algorithms. In: Proceedings of the IEEE International Congress on Evolutionary Computation, pp. 1316–1323. IEEE, San Francisco (2006)
10. Isard, M., Blake, A.: Condensation – Conditional density propagation for visual tracking. Int. Journal of Computer Vision 29(1), 5–28 (1998)
11. Johnson, M.P., Maes, P., Darrell, T.: Evolving visual routines. In: Brooks, R.A., Maes, P. (eds.) Artificial Life IV, Proc. of the 4th Int. Workshop on the Synthesis and Sim. of Living Systems, pp. 198–209. The MIT Press, Cambridge (1994)
12. Katz, A.J., Thrift, P.R.: Generating image filters for target recognition by genetic learning. IEEE Trans. on Pattern Analysis and Machine Int. 16(9), 906–910 (1994)
13. Koza, J.R.: Genetic Programming. On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
14. Krawiec, K., Bhanu, B.: Visual learning by evolutionary and coevolutionary feature synthesis. IEEE Trans. on Evolutionary Computation 11(5), 635–650 (2007)
15. Lohmann, R.: Bionische Verfahren zur Entwicklung visueller Systeme. Ph.D. thesis, Technische Universität Berlin, Verfahrenstechnik und Energietechnik (1991)
16. Miller, J.F.: An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1135–1142. Morgan Kaufmann, San Francisco (1999)
17. Mussi, L., Cagnoni, S.: Artificial creatures for object tracking and segmentation. 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. 255–264. Springer, Heidelberg (2008)
18. NVIDIA: CUDA. Compute Unified Device Architecture. Version 1.1 (2007)
19. Poli, R.: Genetic programming for image analysis. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming, Proc. of the 1st Annual Conf., Stanford University, pp. 363–368. The MIT Press, Cambridge (1996)
20. Rizki, M.M., Tamburino, L.A., Zmuda, M.A.: Evolving multi-resolution feature-detectors. In: Fogel, D.B., Atmar, W. (eds.) Proc. of the 2nd Am. Conf. on Evolutionary Programming, pp. 108–118. Evolutionary Programming Society (1993)
21. Rost, R.J.: OpenGL Shading Language, 2nd edn. Addison-Wesley, Upper Saddle River (2006)
22. Trujillo, L., Olague, G.: Synthesis of interest point detectors through genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, WA, pp. 887–894. ACM, New York (2006)

Markerless Multi-view Articulated Pose Estimation Using Adaptive Hierarchical Particle Swarm Optimisation

Spela Ivekovic, Vijay John, and Emanuele Trucco

School of Computing, University of Dundee, Dundee DD1 4HN
{spelaivekovic,vijayjohn,manueltrucco}@computing.dundee.ac.uk

Abstract. In this paper, we present a new adaptive approach to multi-view markerless articulated human body pose estimation from multi-view video sequences, using Particle Swarm Optimisation (PSO). We address the computational complexity of the recently developed hierarchical PSO (HPSO) approach, which successfully estimated a wide range of different motion with a fixed set of parameters, but incurred an unnecessary overhead in computational complexity. Our adaptive approach, called APSO, preserves the black-box property of the HPSO in that it requires no parameter value input from the user. Instead, it adaptively changes the value of the search parameters online, depending on the quality of the pose estimate in the preceding frame of the sequence. We experimentally compare our adaptive approach with HPSO on four different video sequences and show that the computational complexity can be reduced without sacrificing accuracy and without requiring any user input or prior knowledge about the estimated motion type.

1 Introduction

Video-based markerless articulated pose estimation is an important problem in computer vision which has been given much attention recently [11,16]. Established commercial systems for accurate articulated pose estimation, e.g., Vicon [22], require subjects to wear tight Lycra suits and optical or magnetic markers, an expensive, intrusive and time-consuming solution [16]. Video-based markerless motion capture promises an unintrusive, cheaper and less time-consuming alternative for articulated motion capture.

If the markerless pose estimation is to become truly useful in practice, a black-box solution is necessary which won't require the user to have a detailed knowledge of its internal structure and parameter values. From the user's perspective, regardless of the type of articulated motion being estimated, the algorithm should accept a video sequence as the input and produce a high-quality articulated pose estimate as the output.

In this paper, we present such a black-box approach to articulated pose estimation from multi-view video sequences. We use a powerful global optimisation approach, called particle swarm optimisation (PSO), which has been shown to outperform other search methods (e.g., simulated annealing) on large, difficult and non-linear optimisation problems [7]. John *et al.* [8] use the PSO framework to formulate the articulated pose estimation as a hierarchical search in a constrained search space. Their approach, called HPSO, works as a black box in the sense that it generalises well to different types

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 241–250, 2010.

of motion with fixed PSO parameter settings. However, this ability comes at the price of unnecessarily large computational complexity (although still smaller than the competing techniques). In this paper, we present an adaptive extension of HPSO, called APSO, designed to reduce the search complexity of the HPSO approach without affecting its black-box nature.

This paper is organised as follows. We begin with a discussion of recent related work in Section 2. We describe the PSO algorithm in Section 3, followed by a description of the body model, PSO parametrisation and fitness function in Section 4. The HPSO approach is described in Section 5, and the adaptive extension, APSO, in Section 6. We quantitatively compare our approach with HPSO on several multi-view sequences in Section 7 and conclude in Section 8.

2 Related Work

The literature on markerless human body motion capture is large; recent surveys are [11,16]. Here, we discuss the recent work with respect to the use of motion models and search algorithms.

Motion models. The key motivation behind using motion models is to reduce the dimensionality of an otherwise very expensive or unfeasible search problem. We can regard motion models for human motion tracking as instantaneous or global. *Instantaneous models* predict pose on a frame-by-frame basis; examples are Kalman filtering approaches [10] and particle filtering and variations [2,5,23]. *Global models* seek to describe whole actions (e.g., walking, sitting down) [4,17,14] to provide a context strongly constraining the next pose. The price is reduced generality, as tracking becomes specific to a dictionary of pre-determined actions. Recently, solutions have been proposed which make use of global optimisation to remove the dependency on the pre-trained motion models [8,6], which is also the research direction this paper pursues.

Search. Nearly invariably, pose estimation or tracking is cast as a search in a high-dimensional parameter space, so that an efficient optimiser is of paramount importance. In addition to statistical estimation of dynamic systems (e.g., Kalman and particle filtering), search algorithms reported include iterative closest point (ICP) and variations [12], constrained non-rigid factorization [19], Markov models [14] and gradient boosting [3]. In terms of evolutionary approaches, articulated pose estimation from video sequences has been reported with genetic algorithms [13,24], particle swarm optimisation [7,8,18], and indirectly from voxel data using evolutionary algorithms [20].

In this paper, we extend the pose estimation work recently reported by John *et al.* [8], primarily addressing the computational complexity of the black-box hierarchical PSO search proposed in [8].

3 Particle Swarm Optimisation

PSO is a swarm intelligence technique introduced by Kennedy and Eberhart [9]. The original PSO algorithm has since been modified by several researchers to improve its search capabilities and convergence properties. In this paper, we use the PSO algorithm with an inertia weight parameter, introduced by Shi and Eberhart [21].

3.1 PSO Algorithm with Inertia Weight Parameter

Assume an n-dimensional search space $\mathbb{S} \subseteq \mathbb{R}^n$, a swarm consisting of N particles, each particle representing a candidate solution to the search problem, and a fitness function $f : \mathbb{S} \to \mathbb{R}$ defined on the search space. The i-th particle is represented as an n-dimensional vector $\mathbf{x}^i = (x_1, x_2, ..., x_n)^T \in \mathbb{S}$. The velocity of this particle is also an n-dimensional vector $\mathbf{v}^i = (v_1, v_2, ..., v_n)^T \in \mathbb{S}$. The best position encountered by the i-th particle so far (*personal* best) is denoted by $\mathbf{p}^i = (p_1, p_2, ..., p_n)^T \in \mathbb{S}$ and the value of the fitness function at that position $pbest^i = f(\mathbf{p}^i)$. The index of the particle with the overall best position so far (*global* best) is denoted by g and $gbest = f(\mathbf{p}^g)$. The PSO algorithm with inertia weight can then be stated as follows:

1. **Initialisation:**
   - Initialise a population of particles $\{\mathbf{x}^i\}, i = 1 \dots N$, with random positions and velocities in the search space $\mathbb{S}$. For each particle evaluate the desired fitness function f and set $pbest^i = f(\mathbf{x}^i)$. Identify the best particle in the swarm and store its index as g and its position as $\mathbf{p}^g$.
2. **Repeat** until stopping criterion (see below) is satisfied:
   - Move the swarm by updating the position of every particle $\mathbf{x}^i$ according to

$$\begin{aligned} \mathbf{v}^i_{t+1} &= w\mathbf{v}^i_t + \varphi_1(\mathbf{p}^i_t - \mathbf{x}^i_t) + \varphi_2(\mathbf{p}^g_t - \mathbf{x}^i_t) \\ \mathbf{x}^i_{t+1} &= \mathbf{x}^i_t + \mathbf{v}^i_{t+1} \end{aligned} \tag{1}$$

     where subscript t denotes the time step (iteration) and φ_1, φ_2 are defined below.
   - For $i = 1 \dots N$ update $\mathbf{p}^i$, $pbest^i$, $\mathbf{p}^g$ and $gbest$.

The usual stopping criterion is either that the maximum number of iterations is reached or that the $gbest$ improvement in subsequent iterations becomes small enough. The parameter w is the *inertia weight*. The parameters $\varphi_1 = c_1 rand_1()$ and $\varphi_2 = c_2 rand_2()$, where c_1, c_2 are constant and $rand()$ is a random number drawn from $[0, 1]$, influence the *social* and *cognition* components of the swarm behaviour, respectively. In line with [9], we set $c_1 = c_2 = 2$, which gives the stochastic factor a mean of 1.0 and causes the particles to "overfly" the target about half of the time, while also giving equal importance to both social and cognition components.

The Inertia Weight. We model the inertia change over time with an exponential function which allows us to use a constant sampling step α to gradually guide the swarm from a global to more local exploration:

$$w(\alpha) = \frac{A}{e^{\alpha}}, \quad \alpha \in [0, \ln(10A)], \tag{2}$$

where A denotes the starting value of w, when the sampling variable is $\alpha = 0$. The step α is incremented by a constant $\Delta\alpha = ln(10A)/C$, where C is the chosen number of inertia weight changes per search. The optimisation ends when $w(\alpha)$ falls below 0.1.

4 Body Model, PSO Parametrisation and Fitness Function

To enable performance comparison, we use the same body model and fitness function as the HPSO technique and originally proposed by Balan et al. [1]. In this section, we provide a short summary for completeness and refer the reader to [8,1] for details.

Table 1. Body model joints and their corresponding DOF. There are 31 DOF in total.

JOINT (index)	#	DOF	JOINT (index)	#	DOF
Global body position (1)	3	r_x, r_y, r_z	Right shoulder orientation (7)	3	$\alpha_x^7, \beta_y^7, \gamma_z^7$
Global body orientation (1)	3	$\alpha_x^1, \beta_y^1, \gamma_z^1$	Right elbow orientation (8)	1	β_y^8
Torso orientation (2)	2	β_y^2, γ_z^2	Head orientation (9)	3	$\alpha_x^9, \beta_y^9, \gamma_z^9$
Left clavicle orientation (3)	2	α_x^3, β_y^3	Left hip orientation (10)	3	$\alpha_x^{10}, \beta_y^{10}, \gamma_z^{10}$
Left shoulder orientation (4)	3	$\alpha_x^4, \beta_y^4, \gamma_z^4$	Left knee orientation (11)	1	β_y^{11}
Left elbow orientation (5)	1	β_y^5	Right hip orientation (12)	3	$\alpha_x^{12}, \beta_y^{12}, \gamma_z^{12}$
Right clavicle orientation (6)	2	α_x^6, β_y^6	Right knee orientation (13)	1	β_y^{13}

4.1 Body Model

The human body shape is modelled as a collection of truncated cones (Figure 1(a)). The underlying articulated motion is modelled with a kinematic tree containing 13 nodes, each node corresponding to a specific body joint. For illustration, the indexed joints are shown overlaid on the test subject in Figure 1(b). Every node can have up to 3 rotational degrees of freedom (DOF), while the root node also has 3 translational DOF. In our model, we use a total of 31 DOF, detailed in Table 1.

4.2 PSO Parametrisation

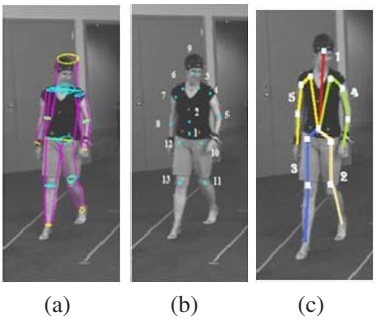
The PSO particle position vector represents an articulated body pose and hence consists of 31 parameters corresponding to the 31 DOF in Table 1:

$$\mathbf{x}^i = (r_x, r_y, r_z, \alpha_x^1, \beta_y^1, \gamma_z^1, \ldots, \beta_y^{13}). \tag{3}$$

4.3 Fitness Function

The fitness function $f(\mathbf{x}^i)$ measures how well a candidate pose $\mathbf{x}^i$ matches the pose of the person in the sequence. It consists of two parts, an edge-based part and a silhouette-based part:

$$f(\mathbf{x}^i) = MSE_{edge}(\mathbf{x}^i) + MSE_{silhouette}(\mathbf{x}^i), \tag{4}$$



(a) (b) (c)

Fig. 1. (a) The truncated-cone body model. (b) Joint positions. (c) Kinematic tree.

where MSE denotes the mean-square error. The edge-based part penalises the distance between the projections of truncated cone edges and the edges in the edge maps obtained from the input images. In the silhouette-based part, a predefined number of points on the surface of the articulated model is projected into the silhouette images and the overlap estimated.

5 The Hierarchical PSO

The HPSO algorithm by John *et al.* [8] splits the 31-dimensional search space into 12 disjoint subspaces which are searched in a pre-defined hierarchical sequence dictated by the hierarchical structure of the kinematic tree representing the human body motion. The subspaces are defined in such a way that only one limb segment at a time is being optimised (see Table 2). Formulating the search in this way significantly reduces its complexity.

Table 2. 12 hierarchical steps of HPSO (*Cf.* Table 1)

(Step 1)	Body position 3 DOF: r_x, r_y, r_z	**(Step 5)**	Left lower arm 2 DOF: γ_z^4, β_y^5	**(Step 9)**	Left upper leg 2 DOF: $\alpha_x^{10}, \beta_y^{10}$
(Step 2)	Body orientation 3 DOF: $\alpha_x^1, \beta_y^1, \gamma_z^1$	**(Step 6)**	Right upper arm 4 DOF: $\alpha_x^6, \beta_y^6, \alpha_x^7, \beta_y^7$	**(Step 10)**	Left lower leg 2 DOF: $\gamma_z^{10}, \beta_y^{11}$
(Step 3)	Torso 2 DOF: β_y^2, γ_z^2	**(Step 7)**	Right lower arm 2 DOF: γ_z^7, β_y^8	**(Step 11)**	Right upper leg 2 DOF: $\alpha_x^{12}, \beta_y^{12}$
(Step 4)	Left upper arm 4 DOF: $\alpha_x^3, \beta_y^3, \alpha_x^4, \beta_y^4$	**(Step 8)**	Head 3 DOF: $\alpha_x^9, \beta_y^9, \gamma_z^9$	**(Step 12)**	Right lower leg 2 DOF: $\gamma_z^{12}, \beta_y^{13}$

The HPSO algorithm is designed to be used as a black box, that is, the user is not required to tweak the search parameters in order to customise the search for a particular type of motion. Instead, the parameter values are set in a way that guarantees that a very wide range of motion, for example, walk, jog, kick, jump, etc. can be estimated without requiring any adjustments. The range of motion that can be successfully estimated depends on the value of the inertia parameter - the higher the starting inertia value (A in Equation (2)), the more agile the motion can be.

6 The Adaptive PSO

When the range of motion we want to estimate with the same parameter settings is very wide, for example, from a simple slow walk to a fast karate kick, the easy solution is to set the starting inertia value A high enough to guarantee that the exploration (rather than exploitation) is given sufficient priority and therefore the fastest motion will be estimated reliably. While the high inertia value is indeed necessary for sequences with fast and sudden motion, it is excessive in sequences where the subject is only walking. In such slow sequences, the high starting inertia value introduces an unnecessary computational overhead. To address this inconsistency, we formulate an adaptive extension of the HPSO approach, the APSO, where the starting inertia value A is adjusted on a frame-by-frame basis.

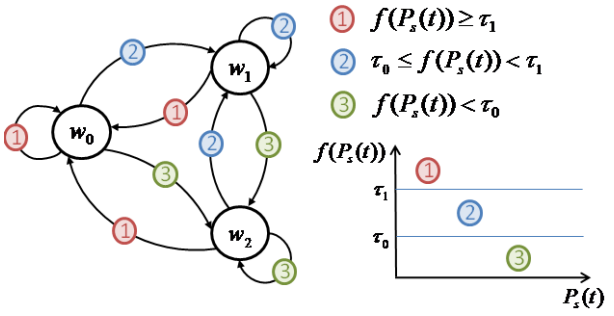


Fig. 2. Adaptive inertia state transition diagram for step s in the hierarchy. At the end of the search, the best pose estimate $P_s(t)$ is evaluated against two fitness function thresholds, τ_0 and τ_1. The higher the $f(P_s(t))$, the better the pose estimate and the smaller the starting inertia for this hierarchical step in the next frame.

6.1 APSO Algorithm

In order to adjust the value of A automatically, online and without user interference, the adjustment process must exploit the information about the search performance in the preceding frame. The APSO approach therefore adaptively changes the next-frame *starting* inertia value for *every hierarchical step* in Table 2 by making use of two quality thresholds, τ_0 and τ_1: when the pose estimate $P_s(t)$ for a hierarchical step s in the current frame is evaluated as good, $f(P_s(t)) \geq \tau_1$, where f is the fitness function, the search region in the next frame is kept small ($A_s^{t+1} = w_0$) as the search is thought to be on target; when the pose estimate is very bad, $f(P_s(t)) < \tau_0$, the search is losing the target and hence the search region in the next frame is expanded significantly ($A_s^{t+1} = w_2$). When the estimate is average, $\tau_0 \leq f(P_s(t)) < \tau_1$, the search region is expanded moderately ($A_s^{t+1} = w_1$), where $w_0 < w_1 < w_2$. The process of adaptively changing the inertia value is illustrated with a state transition diagram in Figure 2.

The adaptive inertia scheme is also used to correct for the effects of an occasional bad starting inertia proposal. For example, in a case where the search is on target in frame t, $f(P_s(t)) \geq \tau_1$, and hence w_0 is proposed for the search in frame $t+1$, but from frame t to frame $t+1$ a sudden, large motion occurs, for which w_0 is not sufficient. We deal with this case as follows. After the search for a particular hierarchical step has been completed, we check the quality of the final pose estimate. If the estimate is bad or average, $f(P_s(t)) < \tau_1$ and the proposed starting inertia value that was used was not the highest inertia value available, $A_s^t = w_i, i < 2$, then the starting inertia value is increased to the next higher value, $A_s^t = w_{i+1}$ and the search for this hierarchical step is repeated. The process repeats until *either* the highest inertia value has been reached, $A_s^t = w_2$, *or* the pose estimate is sufficiently good, $f(P_s(t)) \geq \tau_1$. The value A_s^{t+1} for the next frame is then determined the same way as described in the previous paragraph and illustrated in Figure 2.

The rationale behind the use of this adaptive scheme is in the observation that even fast and sudden actions like, for example, karate kick, consist of segments with slow, medium and fast motion, and therefore searching with the highest inertia value in

every frame would be excessive. The adaptive scheme favours a smaller inertia weight and as the experimental results in Section 7 demonstrate, this is not a bad assumption; the search time indeed decreases in comparison with HPSO without sacrificing the accuracy of the estimates. In fact, given the stochastic nature of the PSO, in our limited experimental evaluation the accuracy actually slightly increases owing to the search repeat strategy which corrects for bad starting values. Making the starting inertia value dependent on the quality of the pose estimate very effectively prevents the search from losing the target and ensures that even very erratic and sudden motion can be followed without diverging.

6.2 Setting τ_0 and τ_1

As a first attempt, we determined the values for τ_0 and τ_1 from a video sequence accompanied with ground truth optical motion capture data. The ground truth poses were used to evaluate the fitness function over a 200-frame sequence and the highest and lowest value of the fitness function were recorded. The interval between the highest and lowest value was then split into three identical bands and the boundaries of the middle band were used as τ_0 and τ_1. As we show with the experimental results, specifying τ_0 and τ_1 in this way does improve the efficiency of the pose estimation, however, we must stress that this is by no means the final solution. Further research is necessary to find a principled way of setting these thresholds which will allow an optimal choice of the search region for every frame of the sequence.

7 Experiments

In this section we compare the performance of the proposed APSO algorithm with that of HPSO.

Datasets. In our experiments, we used 4 datasets: the *Lee walk* sequence included in the Brown University evaluation software [1] and 3 datasets courtesy of the University of

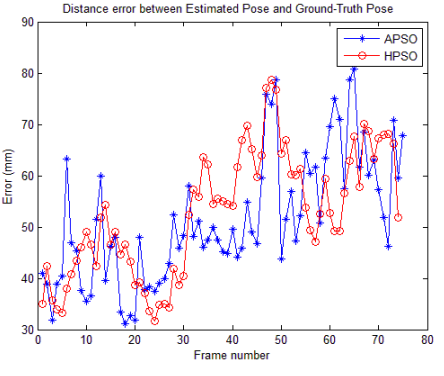


Fig. 3. The error graph on the Lee Walk 30fps sequence

Table 3. Lee Walk sequence: the mean and standard deviation of the distance from the ground truth

Sequence	HPSO ($\mu \pm \sigma$)	Avg time (5 trials)	APSO ($\mu \pm \sigma$)	Avg time (5 trials)
Lee Walk 30Hz	52.5±11.7mm	1 hr,35min	50.8±10.4mm	1 hr, 5min

Table 4. The silhouette/edge overlap measure for the Surrey sequence. Bigger number means better performance.

Sequence	HPSO Mean ± Std.dev	Avg time (5 trials)	APSO Mean ± Std.dev	Avg time (5 trials)
Jon Walk	1.38±0.01	2hr,30min	1.39±0.01	2hr,15min
Tony Kick	1.29±0.02	1hr,30min	1.31±0.01	1hr,15min
Tony Punch	1.32±0.01	1hr,30min	1.34±0.01	1hr,15min

Surrey: *Jon walk*, *Tony kick* and *Tony punch*. The *Lee walk* dataset was captured with 4 synchronised grayscale cameras with resolution 640×480 at 60 fps and came with the ground truth articulated motion data acquired by a Vicon system. The Surrey sequences were acquired by 10 synchronised colour cameras with resolution 720×576 at 25 fps. The test sequences were chosen to cover a range of different body motions.

HPSO and APSO setup. In [8] HPSO was run with only 10 particles; the starting inertia weight was set to $A = 2$, guaranteeing that the particles visit the entire search space, and the stopping inertia was fixed at $w = 0.1$ for all sequences. This amounted to 60 PSO iterations per hierarchical step in HPSO or 7200 fitness function evaluations per frame. In order to enable a meaningful comparison, APSO was also run with only 10 particles, the starting inertia values were set to $w_0 = 0.5$, $w_1 = 1.2$ and $w_2 = 2.0$, the stopping inertia was fixed at 0.1 and pose estimate accuracy thresholds τ_0, τ_1 were derived from the ground-truth pose estimates of the *Lee walk* sequence for every hierarchical step.

7.1 Comparison of APSO vs. HPSO

Lee Walk Results. We tested on a downsampled frame rate of 30 fps instead of the original 60 fps. The results are shown in Table 3 and Figure 3 and indicate that APSO uses less time while also maintaining the average accuracy of the estimation. Table 3 shows the error calculated as the distance between the ground-truth joint values and the values from the pose estimated in each frame. As the algorithm is stochastic in nature, the results shown are averaged over 5 trials. A larger number of trials would provide a better insight, however due to the computational complexity of the algorithm, running a significantly larger number of trials was not practical as HPSO took 70 sec per frame, while APSO varied between 40 sec and 100 sec per frame.

Surrey Results. The Surrey test sequences contain faster motion than the Lee walk sequence. Again, our results for all tested sequences show that APSO reduces the tracking

time. The average overlap and standard deviation for the Surrey sequence over 5 trials are shown in Table 4.

Recovery. John *et al.* [8] remark that HPSO demonstrates the ability to recover from wrong estimates due to error propagation in the hierarchical search within a few frames. APSO, with its search-restart strategy, in fact encounters the error propagation problem a lot less frequently, as any potentially stray estimates are immediately corrected. That explains the slight improvement in APSO estimation accuracy when compared to HPSO.

8 Discussion

Markerless pose estimation from multi-view video sequences is an important problem in computer vision. For any solution to become a useful motion capture tool in practical applications, algorithms are required which can take the burden of parameter tuning and deep algorithmic knowledge away from the intended end-user and work well on a variety of input sequences. A step in the direction of developing such a black-box pose estimation approach was made recently by John *et al.* [8], however, in guaranteeing the black-box property, the HPSO algorithm incurred an unnecessary overhead in computational complexity. In this paper, we presented an adaptive extension of the HPSO approach which reduces the computational complexity of the search. We experimentally demonstrated that the proposed algorithm is computationally more efficient and can still be used on a range of different motions without requiring any parameter tuning by the user.

The adaptive scheme presented in this paper relies only on the information about the quality of the pose estimate in the preceding frame. It does not take into account the information about the speed of the estimated motion which can be extracted from the preceding estimates online, during search. A k-th order autoregressive model trained online to predict the required inertia value based on the estimates in the previous k frames would further improve the efficiency of the search as it would reduce the number of search-restart events which happen due to bad predictions. The choice of inertia values w_0, w_1, w_2 can also be made more principled on the basis of recent work by Poli [15]. We will investigate these improvements in future work.

References

1. Balan, A.O., Sigal, L., Black, M.J.: A quantitative evaluation of video-based 3d person tracking. In: ICCCN 2005, pp. 349–356 (2005)
2. Balan, A.O., Sigal, L., Black, M.J., Davis, J.E., Haussecker, H.W.: Detailed human shape and pose from images. In: CVPR 2007 (2007)
3. Bissacco, A., Yang, M.H., Soatto, S.: Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In: CVPR 2007 (2007)
4. Caillette, F., Galata, A., Howard, T.: Real-time 3-d human body tracking using learnt models of behaviour. CVIU 109(2), 112–125 (2008)
5. Deutscher, J., Reid, I.: Articulated body motion capture by stochastic search. IJCV 61(2) (2005)

6. Gall, J., Rosenhahn, B., Brox, T., Seidel, H.P.: Optimization and filtering for human motion capture - a multi-layer framework. IJCV (online first) (2008)
7. Ivekovic, S., Trucco, E., Petillot, Y.: Human body pose estimation with particle swarm optimisation. Evolutionary Computation 16(4) (2008)
8. John, V., Ivekovic, S., Trucco, E.: Articulated human tracking using HPSO. In: Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP), vol. 1, pp. 531–538 (2009)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE ICNN, vol. 4, pp. 1942–1948 (1995)
10. Mikic, I., Trivedi, M., Hunter, E., Cosman, P.: Human body model acquisition and tracking using voxel data. IJCV 53(3), 199–223 (2003)
11. Moeslund, T.B., Hilton, A., Krueger, V.: A survey of advances in vision-based human motion capture and analysis. CVIU 104(2-3), 90–126 (2006)
12. Muendermann, L., Corazza, S., Andriacchi, T.P.: Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In: CVPR 2007 (2007)
13. Ohya, J., Kishino, F.: Human posture estimation from multiple images using genetic algorithm. In: ICPR, vol. 1, pp. 750–753 (1994)
14. Peursum, P., Venkatesh, S., West, G.: Tracking-as-recognition for articulated full-body human motion analysis. In: CVPR 2007 (2007)
15. Poli, R.: Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. IEEE Transactions on Evolutionary Computation 13(3), 1–10 (2009)
16. Poppe, R.: Vision-based human motion analysis. CVIU 108(1-2), 4–18 (2007)
17. Rosenhahn, B., Brox, T., Seidel, H.P.: Scaled motion dynamics for markerless motion capture. In: CVPR (2007)
18. Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.D.: Parallel global optimization with the particle swarm algorithm. International Journal for Numerical Methods in Engineering 61(13) (2004)
19. Shaji, A., Siddiquie, B., Chandran, S., Suter, D.: Human pose extraction from monocular videos using constrained non-rigid factorization. In: BMVC (2008)
20. Shen, S., Deng, H., Liu, Y.: Probability evolutionary algorithm based human motion tracking using voxel data. In: Proceedings of IEEE CEC 2008, pp. 44–49 (2008)
21. Shi, Y.H., Eberhart, R.C.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation, pp. 69–73 (1998)
22. Vicon: Motion capture systems. (November 2008), http://www.vicon.com
23. Wang, P., Rehg, J.M.: A modular approach to the analysis and evaluation of particle filters for figure tracking. In: CVPR 2006, vol. 1, pp. 790–797 (2006)
24. Zhao, X., Liu, Y.: Generative tracking of 3d human motion by hierarchical annealed genetic algorithm. Pattern Recognition 41(8), 2470–2483 (2008)

Hand Posture Recognition Using Real-Time Artificial Evolution

Benoit Kaufmann[1], Jean Louchet[2], and Evelyne Lutton[1]

[1] INRIA Saclay, Parc Orsay Université, 4 rue Jacques Monod, 91893 Orsay Cedex
benoit.kaufmann@gmail.com, evelyne.lutton@inria.fr
[2] ARTENIA, 24 rue Gay Lussac, 92320 Chatillon
jean.louchet@gmail.com

Abstract. In this paper, we present a hand posture recognition system (configuration and position) we designed as part of a gestural man-machine interface. After a simple image preprocessing, the parameter space (corresponding to the configuration and spatial position of the user's hand) is directly explored using a population of points evolved via an Evolution Strategy. Giving the priority to exploring the parameter space rather than the image, is an alternative to the classical generalisation of the Hough Transform and allows to meet the real-time constraints of the project. The application is an Augmented Reality prototype for a long term exhibition at the Cité des Sciences, Paris. As it will be open to the general public, rather than using conventional peripherals like a mouse or a joystick, a more natural interface has been chosen, using a microcamera embedded into virtual reality goggles in order to exploit the images of the user's hand as input data and enable the user to manipulate virtual objects without any specific training.

1 Introduction

The work described in this paper is part of the REVES project[1], whose aim is to create a prototype for a permanent exhibition at the *Cité des Sciences et de l'Industrie de Paris*[2], called *«Objectifs Terre : La Révolution des satellites»*[3] (objective: Earth). One of the components of this exhibition is devoted to artificial satellites and includes a virtual reality device that enables the user to interact with them.

The user's interface is based on the recognition of hand postures, using a fast generalised Hough transform operated using artificial evolution. The augmented reality device is described in Section 1.1 and the gestural interface in Section 1.2. We present the evolutionary Hough transform methology in Section 2, then describe two different implementations of hand model determination in Sections 3 et 4. Results are presented in Section 5 and the conclusions in Section 6.

[1] REVES ANR (French National Research Agency) contract No 2160 (2007-2009).
[2] http://www.cite-sciences.fr/english
[3] http://www.cite-sciences.fr/objectifs-terre

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 251–260, 2010.

1.1 The Augmented Reality Device

The public will be staying around a large table. Above the table is a videoglobe[4] which displays an animated image of the Earth. Each user is provided with augmented reality "see through" goggles[5] (see figure 1) which allow to display the satellites that turn over the globe, or two- or three-dimensional information when the user is looking down at the table. To this end, the goggles are fitted with a miniature camera. Processing the images from this camera allows to position the synthetic images (e.g. the satellite images) displayed by the goggles relative to what the user can directly see through the goggles. It also allows the user to manipulate the objects displayed, by analysing the user's hand gestures.

Our contribution to this project was to design the algorithms that allow real-time recogniton of hand posture and position from the embedded camera and enable the user to interact with the virtual objects.

The googles. Example of usage.

Fig. 1. The interactive device (courtesy Zile Liu, Laster Technologies)

1.2 The Gestural Interface

In order to ensure natural interaction and allow the user to move whithin the available multimedia content, the interface should provide at least an object designation tool. The object may be e.g. a 3D object, an hypertext link, an icon or a menu entry. Selecting it allows to visualise richer information or move into different menu options. As this is assumed to replace the usual mouse click, we had to devise a simple communication language based on hand configuration.

In addition to this, it is often useful to create a pointer which indicates where the system did locate the user's hand, in order to correct the eye-to-camera parallax and reduce the uncertainty that goes with the size of the detected patterns (e.g. pointed index or the whole hand). Then it is necessary to define two distinct gestures: one to move the pointer, and one to activate the object that has been pointed. The object is selected by changing the hand configuration. The advantage of the latter is the possibility to implement extra primitives such as rollover which allow access to other pieces of information by changing the pointer shape or the object colour. It is also possible to implement commands that move the display zone, or rotate a 3-D object.

[4] Developed by Agenium (http://www.agenium.eu/sections.php?op=listarticles&secid=40)
[5] Developed by Laster Technologies (http://www.laster.fr/?lg=en)

This could have been made possible through using the mouse buttons: a long pressure would switch from one shape to another one, and releasing the button would get back to the first shape. In fact we implemented this functionality by interpreting changes in the user's hand posture: closing and opening the hand is given the same semantics as pressing and releasing a mouse button. This improves the user's feeling to actually interact physically and grasp the object in order to turn or move it. Other display parameters (e.g. the scale) may be edited through creating cursors and moving them a similar way. It is also possible to relate the scale change to the apparent scale of the hand: a grasp plus a movement toward the user corresponds to zooming in, and reversely for zooming out.

The major problem to be solved here is the robust and real time detection of several hand gestures (at least, an open and a closed hand), without prior calibration or learning. To this end we developed an evolutionary Hough transform.

There exists an abundant literature about hand detection, but most of the proposed methods were not adapted to our conditions of use: for instance some methods are based on the detection of the entire body[4,9,3], or at least the user's foreharm[13] to deduce the position of the hands. This solution does not fit with our device because the camera is located on the user's face and therefore can only see the hand and not the rest of the body. Other methods only detect the hand but need a uniform[8] or at least a fixed background[17]. Because the camera we are using is head-mounted, the user may turn their head, which makes the background change and show moving objects that could be wrongly detected as hands and disturb detection. For the same reason, motion-based[6] and multi-camera[16,10] detection methods cannot be applied.

2 EvHough, Evolutionary Exploration of a Parameter Space

The Hough Transform[7] and its classical generalisations[14] fundamentally scan the image looking for a certain type of local features. Each time such a feature has been found, it writes into the parameter space by reversing a model in order to increment the number of votes for the parameter values that are able to provide an explanation to the feature found. As the model is usually not injective, if n is the dimension of the set of parameters, each detected feature generates into the parameter space a variety with dimension $n-1$: a curve if $n=2$, a surface if $n=3$, etc. This results into a very slow process with $n=3$ and unrealistic processing times beyond.

The alternative proposed in [12] then in [11] consists in a direct, smart exploration of the parameter space, using a heuristics given by the Artificial Evolution paradigm. It does not involve any domain-specific knowledge (here, knowledge in image processing) except what has been expressed through the so-called "synthesis model" that allows to calculate the features corresponding to any point in the parameter space.

To this end, the algorithm creates a randomly initialised population of points in the parameter space, then evolves it using genetic operators: selection, mutation, crossover [15,1,5]. The selection process is based on a fitness function which, for each point in the parameter space, executes the model and calculates a value of similarity between the synthesised and the actual feature.

Thus, in the simplest case of sraight line detection, for each pair (ρ, θ), the evolutionary Hough algorithm will calculate the equation of the corresponding line, then evaluate e.g. the number of interest points on this line or the average contrast along it, in order to get the fitness value of the parameter pair (ρ, θ) in question.

The main interest of this is to open the possibility to work efficiently with high dimensional parameter spaces, where a direct Hough-style approach would fail. The main limitation is a classical one in the field of evolutionary algorithms: it remains uneasy to decide when the algorithm has to be terminated. However, in practice this is not a real issue: artificial evolution is naturally time-compliant and, unlike most classical image processing algorithms, may work on data that can be changed during processing. As shown below, this can be an interesting point when processing image sequences in real time.

3 Detection of Hand Models

3.1 Preliminary Processing

The searched models are hand contours shapes, encoded as lists of points, in order to keep the number of points to be tested low. We thus apply a series of filters to the video sequences to be analysed.

Four convolution filters are first used to compute the gradient in 4 directions (vertical, horizontal and two diagonals). For each pixel of the input image, the largest of these 4 values then yields the gradient direction and value. The same information is also kept in memory for each point of the models.

In order to accelerate the computation and facilitate the optimisation process, a proximity function (equation 1) is computed at each frame, and gives for each x of the input image a measurement of its proximity to a contour[2].

$$prox(x) = max_{y \in image}(grad_y / dist_{x,y}) \tag{1}$$

where

- x and y are two pixels of the input image,
- $grad_y$ is the gradient value of pixel y, as defined earlier,
- $dist_{x,y}$ is the distance between x and y according to equation 2[6], with x_i, for $i = 1..8$, the 8 immediate neighbours of x, as defined on figure 2.

$$\begin{aligned} dist_{x,x} &= 1 \\ if \quad x \neq y, \quad dist_{x,y} &= min(dist_{x_1,y} + \sqrt{2}, dist_{x_2,y} + 1, dist_{x_3,y} + \sqrt{2}, \\ & dist_{x_4,y} + 1, dist_{x_5,y} + 1, dist_{x_6,y} + \sqrt{2}, dist_{x_7,y} + 1, dist_{x_8,y} + \sqrt{2}) \end{aligned} \tag{2}$$

The gradient direction for each point of the input image is also updated in order to give the direction of the closest contour point.

In the same time, the colour encoding of the input image (RGB) is transformed to Lab coordinates. In this colorimetric space, a skin colour model has been defined as a

[6] This distance is an approximation of the euclidean distance, that is computed faster.

x_1	x_2	x_3
x_4	x	x_5
x_6	x_7	x_8

Fig. 2. Neighbourhood of x

input image proximity image

Fig. 3. Example of a proximity image

cylinder parallel to the luminance axis whose basis is circular in the space of chrominance (corresponding to an average range of chrominance coordinates of skins). This allows then to compute a low resolution binary image $IndC$, which reads which pixel (or corresponding square area of the input image) may correspond to a skin colour or not. Several cylinders can be defined according to different skin colours, in which case the $IndC$ binary image will indicate if the corresponding pixel area corresponds to at least one of the colour cylinders.

3.2 Genome and Similarity Criterion

The search space, i.e. the space of all possible solutions, is a 5 dimension space, which corresponds to the 5 following parameters:

- $model$ for the hand model (see figure 4)
- tx for the horizontal translation of the model,
- ty for the vertical translation,
- $scale$ for the apparent scale of the model, which varies with the size of the user's hand and its distance to the camera,
- $rotate$ for the rotation of the hand with respect to the optical axis of the camera.

The other rotations are ignored, as we consider that the user designates objects with arms stretched, and hands quasi othogonal to the optical axis. This allow to restrict the search to 2D models of hands.

The similarity criterion (or $fitness$) is a combination of three independent criteria, f_1, f_2 and f_3.

- f_1 gives a measure of the distance between the image and model contours, it is based on the use of the proximity image (see figure 3):

$$f_1(S) = \frac{\sum_{p \in contour} p \times prox(proj(p))}{card(contour)} \tag{3}$$

  $S = [model, tx, ty, scale, rotate]$ is a point of the search space, $proj(p)$ is the projection of p via the transformation defined by S and $card(contour)$ is the number of contour pixels. Dividing by $card(contour)$ allows the criterion to be independent of the number of contour pixels.

– f_2 measures the correpondance of gradient directions between image area and model:

$$f_2(S) = \frac{card(C)}{card(contour)} \tag{4}$$

$card(C)$ is the number of contour pixels x whose direction corresponds to the model's direction, and $card(contour)$ is the number of pixels of the contour.

– f_3 uses the binary image $IndC$ to verify if the interior of models contours has the right colour (this calculation is usually done on a lower scale image):

$$f_3(S) = \frac{card(D)}{card(E)} \tag{5}$$

$card(D)$ is the number of interior pixels of the model whose projection on the image corresponds to "true" on $IndC$ and $card(E)$ is the number of interior pixels of the model. Once again dividing by $card(E)$ allows to have a measurement independent to the size of the model.

The resemblance criterion for a solution S is then:

$$fitness(S) = \frac{\alpha_1 \times f_1(S) + \alpha_2 \times f_2(s) + \alpha_3 \times f_3(S)}{\alpha_1 + \alpha_2 + \alpha_3} \tag{6}$$

In our implementation, we choose $\alpha_1 = 10$, $\alpha_2 = 10$ and $\alpha_3 = 1$.

3.3 Genetic Engine

A steady state approach has been chosen for the EA engine, which means that each new individual immediately replaces an old unadapted one within the population: no generation synchronism is used. The individuals to be replaced are chosen using the tournament technique. Three genetic operators are used:

– *immigration:* a new individual is randomly generated,
– *mutation:* a new individual is a small perturbation of an existing one,
– *crossover:* a new individual is created as a random weighted average of two existing individuals.

After an immigration or a mutation is done, two individuals are chosen in the population and the one who gets the lower fitness is replaced with the new individual (tournament of size 2). After a crossover, a tournament of size 3 is used.

Steady state engine and tournament selection have been chosen in order to better maintain the diversity and limit the risk of premature convergence. This point is actually critical here, as we operate a small sized population in order to obtain real time performance. A generational process would necessitate fitness ranking which is time consuming. For the same reason, a fixed size population is used, in order to minimise time consuming memory allocations.

Additionnally, as this EA runs on an varying environment, the fitness function is recomputed at each new frame. We actually get an asynchronous algorithm, that uses video information at the time it is available.

Finally, to compare with a generational EA, we measure the evolution in terms of "pseudo-generations", which corresponds here to the evaluation of a number of individuals equivalent to a generation gap (i.e. 40% of the population size, see section 5).

4 Additional Improvements

Experiments based on the previous algorithm yield good results, however in some conditions, some model representatives were lost in the population. As a consequence, the detection of a model change was unstable and slower. We thus preferred an encoding that naturally balances the number of representatives within the population, by not explicitly using the $model$ any more: the fitness is then computed for each searched model. Each individual thus has now n fitness values, one for each of the n searched models. The resulting fitness is the maximal value of the n fitness:

$$fitness(i) = max(fitness_1(i), ..., fitness_n(i)) \tag{7}$$

$fitness(i)$ is the fitness value of individual i and $fitness_k(i)$ is the fitness value of individual i according to model k ($k \in [1..n]$), computed using equation 6.

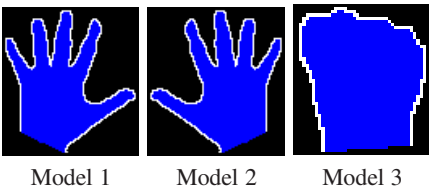
For each frame, the detected model is the one which corresponds to the highest fitness, as soon as it is greater than a fixed threshold (0.3 for the results presented in the next section). To avoid unstabilities due to the definitions of a strict threshold, the detection decision is based on a double threshold and takes into account the model recognised at the previous frame:

- if the largest fitness of the last population of the current frame is larger than threshold T, the corresponding model is detected,
- if the largest fitness is in $[0.7 \times T, T]$, and corresponds to the detected model of the previous frame, it is maintained as "detected,"
- else, nothing is detected.

5 Results

The results presented below have been obtained on a netBook *MSI U100*, equipped with a single core 1,60 GHz *Intel Atom N270* processor, with 1 GB of memory, and using *GNU/Linux Mandriva 2009*. The images are captured with the integrated webcam (resolution of 160 $\times$ 120 pixels).

Examples of detection on a video sequence are displayed on figure 6, and the curves of figure 5 give a representation of the distribution of representatives of each of the searched models within the population. Each graph (one per searched model, see figure 4) presents three curves which are the maximal and average of fitness values,



Model 1 Model 2 Model 3

Fig. 4. Hand models: the white pixels are contours, blue pixels are pixels where the colour model is evaluated, and black pixels are ignored

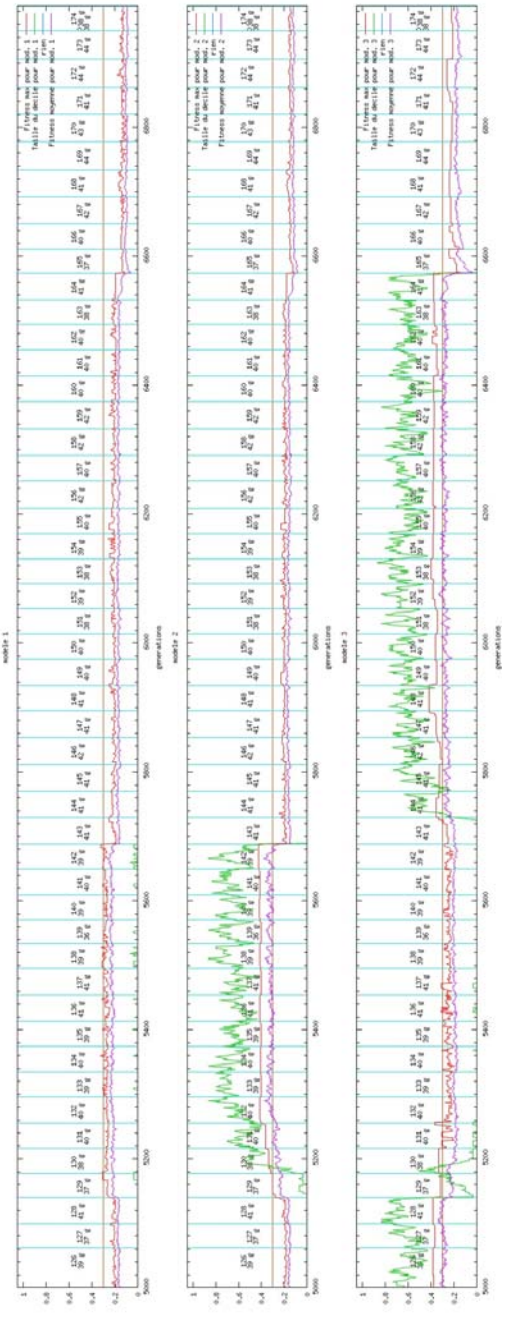


Fig. 5. Variation of the population models distribution and fitness values, for a sequence where the 3 models of figure 4 are searched

Fig. 6. Example of detection on a video sequence in natural light environment. Blue shapes show the detected model, red points correspond to the position (origin of the models) (x, y) of all individuals of the current population.

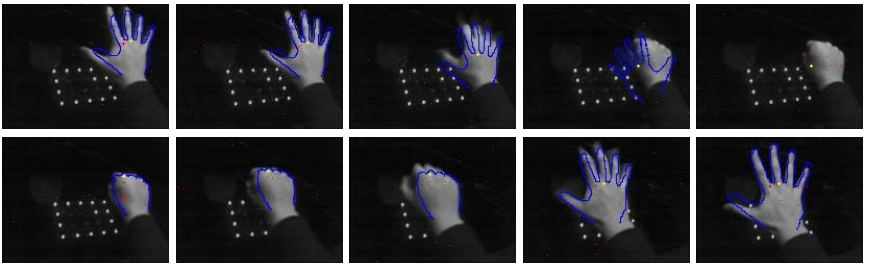


Fig. 7. Example of a detection on a video sequence in controlled light environment. A yellow marker represents the mouse position.

and the proportion of individuals whose fitness is at least equal to 90% of the maximal fitness. The horizontal line represents the model detection threshold (0.3). Vertical lines mark the transition between two successive frames. For each frame, its number, and the number of generations that were run on it, are displayed.

It can be noticed that the curve corresponding to the fitness max of model 1 (which is not present in this part of the video) never crosses the 0.3 threshold. On the contrary, one may notice on graphs of models 2 and 3, an alternance of detection of these models, and for each detection phase, a concentration of the population around the maximal values (green curve).

Other examples of detection are given on figure 7, in the experimental conditions described in section 1.1. A video sequence is also available on `http://apis.saclay.inria.fr/twiki/bin/view/Apis/HandGestureRecognition`

6 Conclusion

We showed that a real-time evolutionary algorithm can be run on a general-public device, using low computation power. The specific components of this evolution algorithm are a small population, an asynchronous steady-state genetic engine, that produces a variable number of generations between two frames, depending on the computation load of the whole system. An implementation on another computer would result in a different average number of generations between two frames. This characteristic yields an efficient capability to the algorithm, as it is able to exploit the available data as soon as they appear. Of course the number of generations between two frames condition the quality of the detection, and its reactivity to sudden changes. In extreme conditions, if the algorithm cannot obtain enough computation power, the population does not have enough time to converge between two frames. From the user's side, this results in late detections, and unability to follow rapid moves.

The genetic encoding finally adopted (section 4) has been preferred for robustness and reactivity reasons. In case the genome carries the model, a delay is necessary in order to let representatives of a new model grow, while in the second version, this change can be immediate. Of course this solution is only efficient if the number of searched models is low, as fitness calculation is longer (n fitness computations for n models).

Additionnally, for the application described in section 1.1, a Kalman filter on the detected parameters for each frame allows to obtain a smoother detection. Finally, the

display of a "mouse pointer" in the virtual environment (yellow mark in figure 7) gives functionalities similar to classical computer mouses.

Acknowledgments. The authors thank Sergio Alejandro Mota-Gutierrez, of Guanajuato University, Mexico, student of the *Electrical Engineering Master Program*, for his contribution to the colour based fitness computation.

References

1. Baeck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolution strategies. In: International Conference on Genetic Algorithms, July 13-16, pp. 2–10 (1991)
2. Borgefors, G.: Distance transformations in arbitrary dimensions. Computer Vision, Graphics, and Image Processing 27, 321–345 (1984)
3. Carbini, S., Viallet, J.E., Bernier, O.: Pointing gesture visual recognition by body feature detection and tracking. Computational Imaging and Vision 32, 203–208 (2006)
4. Darby, J., Li, B., Costen, N.: Activity classification for interactive game interfaces. Int. J. Comput. Games Technol. 2008, 1–7 (2008)
5. Goldberg, D.A.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc., Reading (1989)
6. Harper, R., Rauterberg, M., Combetto, M. (eds.): ICEC 2006. LNCS, vol. 4161. Springer, Heidelberg (2006)
7. Hough, P.V.C.: Methods and means of recognizing complex patterns. Tech. rep., US Patent 3.069.654.18 (December 1962)
8. Ionescu, B., Coquin, D., Lambert, P., Buzuloiu, V.: Dynamic hand gesture recognition using the skeleton of the hand. EURASIP J. Appl. Signal Process. 2005, 2101–2109 (2005)
9. Kim, H.J., Kwak, K.C., Lee, J.: Bimanual Hand Tracking. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3980, pp. 955–963. Springer, Heidelberg (2006)
10. Kirishima, T., Manabe, Y., Sato, K., Chihara, K.: Real-time multiview recognition of human gestures by distributed image processing. J. Image Video Process. 2010, 1–13 (2010)
11. Louchet, J.: From Hough to Darwin: an Individual Evolutionary Strategy applied to Artificial Vision. In: Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M., Ronald, E. (eds.) AE 1999. LNCS, vol. 1829. Springer, Heidelberg (2000)
12. Lutton, E., Martinez, P.: A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images. In: 12-ICPR, Jerusalem, Israel, October 9-13 (1994)
13. Maimon, D., Yeshurun, Y.: Hand Detection by Direct Convexity Estimation. In: Tistarelli, M., Bigun, J., Grosso, E. (eds.) Advanced Studies in Biometrics. LNCS, vol. 3161, pp. 105–113. Springer, Heidelberg (2005)
14. Maître, H.: Un panorama de la transformation de Hough. Traitement du Signal 2(4), 305–317 (1985)
15. Rechenberg, I.: Evolution Strategy : Nature's way of optimization. In: Bergman, H.W. (ed.) Optimization : methods and and applications. Possibilities and Limitations. Lecture Notes in Engineering, vol. 17, pp. 106–126. Springer, Berlin (1989)
16. Stødle, D., Hagen, T.M.S., Bjørndalen, J.M., Anshus, O.J.: Gesture-based, touch-free multi-user gaming on wall-sized, high-resolution tiled displays. Journal of Virtual Reality and Broadcasting 5(10) (November 2008)
17. Zhao, S., Tan, W., Wen, S., Liu, Y.: An improved algorithm of hand gesture recognition under intricate background. In: Xiong, C.-H., Liu, H., Huang, Y., Xiong, Y.L. (eds.) ICIRA 2008. LNCS (LNAI), vol. 5314, pp. 786–794. Springer, Heidelberg (2008)

Comparing Cellular and Panmictic Genetic Algorithms for Real-Time Object Detection

Jesús Martínez-Gómez, José Antonio Gámez, and Ismael García-Varea

Computing Systems Department, SIMD i^3A
University of Castilla-la Mancha, Albacete, Spain
{jesus_martinez,jgamez,ivarea}@dsi.uclm.es

Abstract. Object detection is a key point in robotics, both in localization and robot decision making. Genetic Algorithms (GAs) have proven to work well in this type of tasks, but they usually give rise to heavy computational processes. The scope of this study is the Standard Platform category of the RoboCup soccer competition, and so *real-time* object detection is needed. Because of this, we constraint ourselves to the use of tiny GAs. The main problem with this type of GAs is their premature convergence to local optima. In this paper we study two different approaches to overcoming this problem: the use of population re-starts, and the use of a cellular GA instead of the standard generational one. The combination of these approaches with a clever initialisation of the population has been analyzed experimentally, and from the results we can conclude that for our problem the best choice is the use of cellular GAs.

1 Introduction

For mobile robotics, image processing has become one of the most important elements. Most current proposals have to choose between low execution time and good system performance. This balance becomes a keystone for RoboCup[9] environments, where robotic teams play football matches within controlled fields using specific rules. All the robots have to take real-time decisions using the information retrieved from the environment with their sensors (mainly vision cameras).

Our approach to real-time object detection is the use of genetic algorithms[7] (GAs). According to the definition of this type of algorithms, the individuals will represent the object we want to detect, and the fitness function will represent the quality of the detection process.

In order to develop a real-time system, the number of parameters that govern a GA (number of generations and population size) must be reduced as much as possible. This reduction speeds up the processing but low quality individuals may be obtained. The convergence of a GA with a low number of generations should be fast to obtain high-quality individuals, but that increases the risk of falling into local optima. The negative impact of local optima can be avoided by preventing premature convergence or using strategies to escape from them.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 261–271, 2010.

In this paper we study restart scheduling to escape from local optima and a cellular structure[1] is proposed to prevent premature convergence. Different experiments were carried out to compare panmictic and cellular GAs and to study the impact of restart scheduling over these structures.

This study was mainly inspired by a vision system developed for the RoboCup 2009 soccer competition[6]. That system was developed using a panmictic GA with hard processing time restrictions. In that work, all the experiments were carried out using a population size of 12 individuals and 24 iterations. Under these conditions, GAs can have problems to converge and the population initialization becomes a keystone. In order to guarantee good convergence, the information retrieved from the image processing and from the last n detections is used to initialize the population. In this situation, an individual can be initialized in three different ways: randomly, using the information extracted from the image processing or cloning an individual from a previous population.

The main drawback of this new reasoning is the high risk of local optima, which is increased due to the new population initialization. In order to escape from local optima, different approaches have been proposed[4]. Most of these approaches propose a restart scheduling and that was the option adopted in [6].

2 Vision System

The vision system has to detect and estimate distance and orientation to key elements in a football field. These elements are field lines, opponents, team members, the ball and the goals. Some of these elements have not changed during the last few years (field lines and the ball), and their detections can be performed using fast and simple techniques based on colour filters and scan-lines [10,5,3].

In this paper, we focus on goal detection because the new goals are very different from those used in previous editions (the new size is considerably bigger than the old one). The shape of a goal in a frame depends on the position and orientation between the robot camera and the goal. Therefore, a high number of partial occlusions and observations appear and this makes impossible to use a case-based approach. The ball and the new goal (partially observed) are illustrated in Fig.1, which is a real frame captured by the robot camera.

When using a genetic approach to solve the problem of goal detection, an individual has to represent the detection of the goal g placed at distance d with orientation or. This information is contrasted with that extracted from the last

Fig. 1. Frame showing the ball and a partial observation of the blue goal

frame captured by the robot camera. The fitness will be high for individuals whose information is plausible with respect to the last image.

2.1 Individual Representation

An individual must store the information necessary to evaluate the fitness function. An individual is represented by four genes: $< d, \alpha, \beta, \theta >$. Fig.2 graphically shows three of the parameters to be estimated: d is the distance between camera and goal, and α and β are the differences in orientation in the x-axis and in the y-axis respectively. A third component for the orientation difference in the z-axis is not needed, because by using horizon detection techniques [2] the image can be processed to show all the objects parallel to the floor.

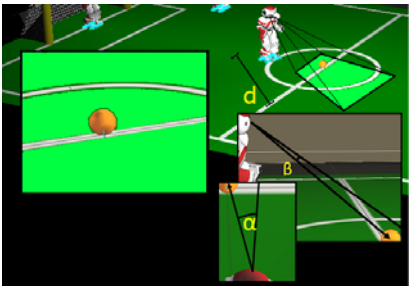


Fig. 2. Meaning of d, α, β

Gen θ represents own goal orientation and its value is limited to between -90 and 90 degrees. All the genes are represented by numerical values, limited by the maximum distance detection for d, and by the field of view for α and β.

2.2 Fitness Function

The fitness function returns numeric values, according to the goodness of the projection obtained with the parameters $< d, \alpha, \beta, \theta >$ of an individual. To evaluate an individual, its genes are translated into the projection of the goal that the individual represents. The projection needs a start position $< x, y >$, obtained from α and β, and the size of the object depends on d.

Fig. 3. Projections obtained for six individuals

Fig.3 shows six projections obtained with six different individuals representing yellow goals. It can be observed how goals which are far from the camera obtain smaller projections, and how some parts of the projections are beyond the frame boundaries.

A goal projection is evaluated by comparing it with the information obtained from the filtering process. All the pixels obtained with the projection are matched with the pixel distribution obtained after applying the yellow or the blue filter (it depends on the goal we are trying to detect). The colour filtering is carried out by defining a top and bottom limit for the YUV colour components. A pixel will successfully pass a filter only if all its components are between these limits. Fig.4 shows a colour (yellow, white and orange) filtering example. Lighting conditions are constant for RoboCup environments and we assume the use of optimal filters.

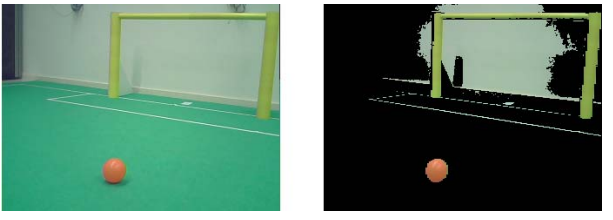


Fig. 4. Colour-filtering process

The fitness of an individual is defined as the minimum value of:

- % of pixels of the projection that have passed the colour filter.
- % of pixels that passed the colour filter and belong to the valid projection pixels.

2.3 General Processing Scheme

The processing starts for every frame captured by the camera of the robot. A new frame will evolve a new population only for a goal (blue and yellow) that is likely to appear in the image. That plausibility is estimated studying the number of pixels that correctly passed the corresponding colour filter. Fig.5 presents the algorithm of the general processing scheme, where the population evolution depends on the type of GA we are using: cellular or panmictic.

```
Capture a new image and filter it with colour filters
for each one of the two goals
    if we have obtained enough pixels
        Evolve a new population
        Apply local search over the best individual
        Return the estimated distance and orientation to the goal
    end if
end for
```

Fig. 5. General system processing scheme

2.4 Population Initialization

The current vision system must work in real-time. Therefore, convergence of the algorithm must be obtained with a small number of generations and individuals. In order to achieve this goal, population initialization is performed using information obtained from two different sources: colour filtering (centroid and number of pixels of the colour distribution) and previous populations. The information obtained from previous populations allows us to take advantage of the high similarity between consecutives frames; that is, if a frame shows the blue/yellow goal at distance d with orientation or, the next frame has a high probability of showing the same goal with similar d and or values.

An individual can now be initialized in three different ways:

- Randomly (R)
- Using the information from the filtering process (F)
- Cloning an individual from a previous population (C)

A probabilistic selection is carried out to select the way in which an individual is initialized. These probabilities depend on the number of frames from the last frame that detected the goal we are studying (denoted as $NFLR$). These probabilities are computed as follows:

$$P(C) = max(0.5 - 0.5 * (NFLR/10), 0)$$
$$P(F) = (1 - (P(C))) * 0.66$$
$$P(R) = (1 - (P(C))) * 0.34$$

If the value of $NFLR$ is bigger than 10, $P(C)$ will be 0 because we assume that after 10 frames, similarity between frames cannot be exploited.

3 Genetic Algorithm Structure

As was mentioned in section 1, the system proposed in [6] was developed using a panmictic GA. The solution adopted to escape from early local optima was to restart the population after a percentage of iterations failed to improve the best fitness.

In this paper, we propose to use an alternative structure named cellular GAs. Using this structure, the crossover between individuals will only be performed if both individuals are similar. Therefore, it is necessary to define a structure for the individuals which is based on their genes. This step can be hard to solve when the genes have nominal values, but not with numeric values.

We propose a structure based on the gene d, which represents the distance between the robot camera and the goal. This gene has numeric values limited to between 30 (minimum distance to perform an object detection) and 721 cm (the football field is 600 x 400 cm). Distance is considered the most important information and therefore the gene d was selected to define the structure. The difference between a panmictic and a cellular structure can be observed in Fig. 6,

where the darkness represents distance values (white for closer goals and black for distant goals). With the panmictic structure, no order is defined for individuals and all crossovers between them are possible. On the other hand, with the cellular structure the population is ordered and a crossover between two individuals can only be performed if they are close together.

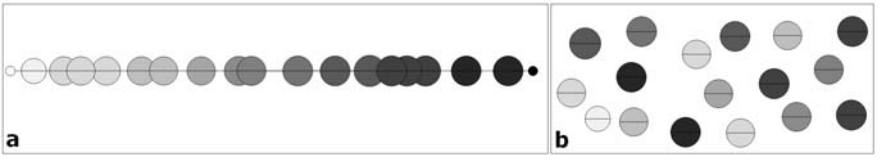


Fig. 6. Cellular (a) and Panmictic (b) Structure

The minimum degree of similarity between two individuals which is necessary to perform the crossover is dynamically defined. If we denote $d_{max_{dif}}$ as the maximum difference for d ($d_{max} - d_{min}$) and n_{ind} as the size of the population, individuals i_a and i_b can only be crossed if the difference between d values of i_a and i_b is smaller than $d_{max_{dif}}/(n_{ind}/2)$. This neighbourhood threshold ($d_{max_{dif}}/(n_{ind}/2)$) is denoted as n_t. In order to illustrate this neighbourhood, Fig. 7 shows two different scenarios where a change in the distribution of 10 individuals causes n_t to vary considerably.

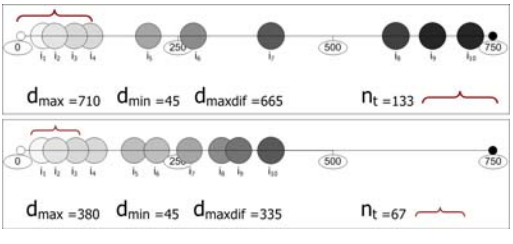


Fig. 7. Different individual distributions and neighbourhood thresholds

It can be observed how the neighbourhood threshold (n_t) is reduced when individuals are close together. For the distribution proposed in the upper image of Fig. 7, the individual i_2 can be crossed with individuals i_1, i_3 and i_4. All these individuals $i_1 ... i_4$ are repeated for the bottom image, but due to the new n_t value, i_2 can only be crossed with i_1 and i_3.

3.1 Algorithm Evolution

In order to highlight the main differences between both structures, we will perform some experiments over a test frame. This test frame contains a partially occluded goal, placed at a distance of 240 cm. These experiments show how the algorithm converges during the iterations. We will use a bubble graph (see Fig.8),

where the x-axis represents the iterations, the y-axis the value for gene d and the size of the bubbles represents the fitness value of the individual. The same experiment with the same parameters was performed for two different structures: panmictic and cellular GA. These experiments were performed with 12 individuals, 24 iterations, 5% mutation probability and one-point as crossover operator. No restarting schedule was applied.

Panmictic GA. Fig. 8a shows the typical evolution for the fitness of 12 individuals over 24 iterations using a panmictic GA. It can be observed that after 5 or 6 iterations, all the individuals have similar values for the gene d, which represents the distance to the goal in centimetres.

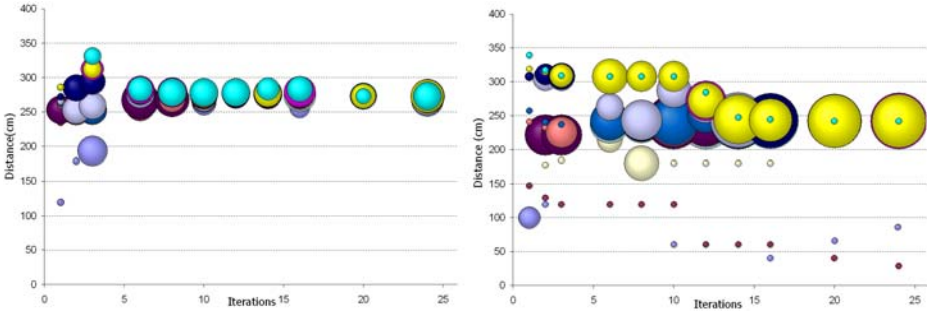


Fig. 8. Individual fitness evolution for a panmictic (a) and a cellular (b) GA. Bubble's diameter represents an individual's fitness.

The best fitness value is not improved upon after iteration 6, where a value of 0.26907 is obtained. At the end of the algorithm most of the individuals have fitness values close to 0.22. The fast convergence of the algorithm into local optima makes increasing the number of iterations pointless. This happens due to the special characteristics of the fitness function. Only individuals really close to the solution obtain fitness values greater than zero. Moreover, small variations in the values of the genes cause enormous variations in the goodness value obtained with the fitness function.

Cellular GA. Fig. 8b shows the evolution for the fitness of 12 individuals over 24 iterations using a cellular structure. In this case, the convergence of the system is slower than that obtained in Fig. 8a. During the iterations, a bigger section of the search space is explored and the best fitness value (0.6527) is obtained at the last iteration. Even when 10 of the individuals converge to the global optima (distance 240) at the last iteration, two individuals explore the search space between distance values of 40 and 80.

The cellular structure allows the algorithm to obtain best fitness values and to escape from local optima. The convergence is slower but better individuals are reached. Fig. 9 shows the evolution of the best fitness for both structures. In this graph, the best fitness gradually increases from the first to the last iteration

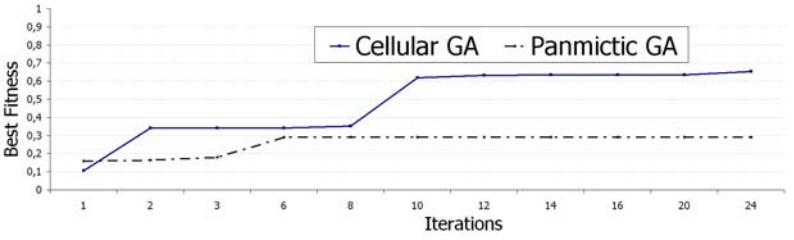


Fig. 9. Best fitness evolution for a panmictic and a cellular structure

for the cellular structure. On the other hand, the panmictic GA obtains the best fitness value at iteration 6, and after this moment no improvements are obtained.

In view of these results, the most appropriate structure for our proposed genetic algorithm is cellular.

4 Experiments and Results

The experiments were carried out on a RoboCup Standard Platform football field, with the official goals, a 6 x 4 metre carpet and a ball. A biped robot Nao[1] was used, taking images in YUV format. While the robot was moving around the environment, the frames captured by the robot's camera were stored. In addition to this information, we also stored the real distance to the yellow and blue goal for each frame. In order to perform a complete comparison of our entire proposal, we used the same test sequence for all the experiments. This test sequence contains frames that captured the blue goal placed between 217 and 103 cm.

Using the real distance to the goals (at the instant when a frame was captured), our algorithm was evaluated by studying the difference between the real and the estimated distance as the error rate. The estimated distance was the value of the d gene of the individual with the highest fitness value. Lighting conditions were stable throughout the experiments, and the colour filters were optimal.

All the experiments were performed using 12 individuals, 24 iterations, 5% mutation probability and one-point as crossover type. No elitism was used and so the entire population was replaced by the offspring at the end of each iteration.

After the population has evolved, a simple local-search process (Hill Climbing) is applied to the best individual. This processing allows us to improve the best fitness. The local search is applied by evaluating positive and negative variations for the genes of the individual. Using concepts and strategies from different metaheuristics, our algorithm should be termed a memetic algorithm [8] instead of a genetic one.

4.1 Experiment 1 - Adding Restart Scheduling

The first experiment has the objective of testing the use of restart scheduling for the panmictic structure. We applied our genetic algorithm using the standard

[1] http://www.aldebaran-robotics.com/eng/Nao.php

parameters described above. Restart scheduling consists of restarting the population after 25% of the iterations have failed to improve the best fitness value. The test sequence (30 frames) was processed 10 times, saving the difference between the real and estimated distance. Table 1 shows the mean absolute error (MAE) for the panmictic GA, with and without restarts. In order to illustrate the robustness of both proposals, the table shows the percentage of frames that obtained an error rate lower than 100, 75, 50 and 25 cm.

Table 1. Mean Absolute Error (MAE) in centimetres and Standard Deviation (Sd) for a panmictic(a) and a cellular(b) genetic algorithm, with and without restarts

Restart Scheduling	MAE(cm)	Standard Deviation	Percentage of frames (in cm) $\leq$ 100 cm	$\leq$ 75 cm	$\leq$ 50 cm	$\leq$ 25 cm
Panmictic GA						
Without restarts	90.79	53.98	53.3%	47.7%	43.7%	34.3%
With restarts	58.53	43.68	76.0%	67.3%	55.7%	40.3%
Cellular GA						
Without restarts	39.21	35.35	89.0%	80.0%	68.3%	50.3%
With restarts	59.39	41.78	76.3%	64.3%	53.7%	32.7%

4.2 Experiment 2 - Testing the Cellular Structure

The objective of the second experiment was to prove that, for our problem, the cellular structure is more appropriate than the panmictic one. We tested two different configurations using a cellular GA, with and without a restarting schedule. For this experiment we used the same test sequence and restarting schedule as those used for the first experiment.

If we compare table 1 (a) and (b) we can see that the robustness and the performance of the algorithm have improved. Mean absolute error for the best configuration (cellular GA without restarts) is lower than 40 cm (in an environment with a maximum distance of 721 cm.) and nearly 70% of frames obtained differences for the distance estimation under 50 cm. Highest error values were obtained at the beginning of each test sequence, as can be observed in Fig. 10.

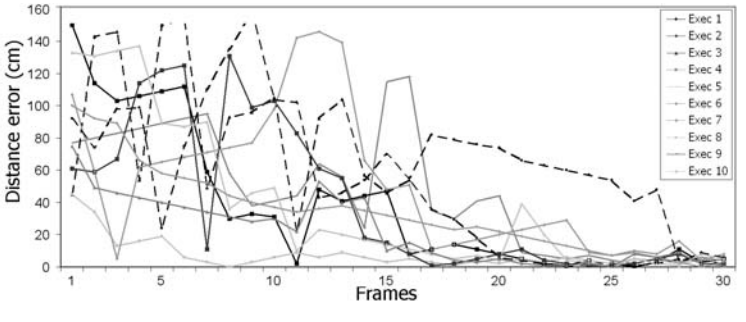


Fig. 10. Error evolution for ten executions of a cellular GA without restarts

After a few frames the error decreased and, specially after the 17th frame, the error was lower than 50 for 9 of the 10 executions.

In contrast to the improvement obtained for the panmictic GA, restart scheduling obtained worse results when using the cellular structure. This is because the cellular structure causes a slower convergence and the restarts (after 45% of iterations fail to improve the best fitness) do not allow the algorithm to achieve the best results.

5 Conclusions and Future Work

In this paper we have dealt with the problem of object detection in real time by using GAs. The use of GAs is justified by the complexity of the environment (which includes occlusions,) which makes it imposible to use a case-based approach. However, because of the real-time requirements, tiny GAs must be used. This type of GAs (few individuals per population and few generations) have a great risk of converging to a local optimum.

We have tested two different approaches: panmitic GAs with restarts and cellular GAs. Of the two options, our empirical results support the argument that cellular GAs get the best results in the problem under study. The *low* error rate obtained for the best configuration tested should allow the system to be used to perform robot self-localization, and this topic is one of the lines for future research that we want to explore.

Acknowledgements

The authors acknowledge the financial support provided by the Spanish "Junta de Comunidades de Castilla-La Mancha (Consejería de Educación y Ciencia)" under PCI08-0048-8577 and PBI-0210-7127 Projects and FEDER funds.

References

1. Alba, E., Dorronsoro, B.: Cellular genetic algorithms, vol. 1 (March 2008)
2. Bach, J., Jungel, M.: Using pattern matching on a flexible, horizon-aligned grid for robotic vision. Concurrency, Specification and Programming-CSP 1(2002), 11–19 (2002)
3. Coath, G., Musumeci, P.: Adaptive arc fitting for ball detection in robocup. In: APRS Workshop on Digital Image Analysing. pp. 63–68 (2003)
4. Fukunaga, A.: Restart scheduling for genetic algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 357–366. Springer, Heidelberg (1998)
5. Jüngel, M., Hoffmann, J., Lötzsch, M.: A real-time auto-adjusting vision system for robotic soccer. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 214–225. Springer, Heidelberg (2004)
6. Martínez-Gómez, J., Gámez, J.A., García-Varea, I., Matellán, V.: Using genetic algorithm for real-time object detection. In: Proceedings of the 2009 RoboCup Symposium, pp. 215–227. Springer, Heidelberg (2009)

7. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1998)
8. Moscato, P.: Memetic algorithms: a short introduction. Mcgraw-Hill'S Advanced Topics in Computer Science Series, pp. 219–234 (1999)
9. Rofer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jungel, M., Laue, T., Lotzsch, M., Nistico, W., Spranger, M.: GermanTeam 2004. Team Report RoboCup (2004)
10. Wasik, Z., Saffiotti, A.: Robust color segmentation for the robocup domain. In: Proc. of the Int. Conf. on Pattern Recognition (ICPR), vol. 2, pp. 651–654 (2002)

Bloat Free Genetic Programming versus Classification Trees for Identification of Burned Areas in Satellite Imagery

Sara Silva[1,2], Maria J. Vasconcelos[3], and Joana B. Melo[3,4]

[1] INESC-ID Lisboa, Portugal
[2] Center for Informatics and Systems of the University of Coimbra, Portugal
[3] Tropical Research Institute, Lisbon, Portugal
[4] Instituto Superior de Agronomia, UTL, Portugal
sara@{kdbio.inesc-id.pt,dei.uc.pt}
{maria.perestrelo,joana.lx.bm}@gmail.com

Abstract. This paper compares Genetic Programming and Classification Trees on a problem of identification of burned areas in satellite imagery. Additionally, it studies how the most recently recognized bloat control technique, Operator Equalisation, affects the quality of the solutions provided by Genetic Programming. The merit of each approach is assessed not only by its classification accuracy, but also by the ability to predict the correctness of its own classifications, and the ability to provide solutions that are human readable and robust to data inaccuracies. The results reveal that both approaches achieve high accuracy with no overfitting, and that Genetic Programming can reveal some surprises and offer interesting advantages even on a simple problem so easily tackled by the popular Classification Trees. Operator Equalisation proved to be crucial.

1 Introduction

Genetic Programming (GP) is the automated learning of computer programs, using Darwinian selection and Mendelian genetics as sources of inspiration [10]. The search space of GP is virtually unlimited and programs tend to grow larger during the evolutionary process. Code growth is a natural result of genetic operators in search of better solutions, but it has been shown that beyond a certain program length the distribution of fitness converges to a limit [7]. Bloat can be defined as an excess of code growth without a corresponding improvement in fitness. Several theories explaining bloat, and many different bloat control methods, have been proposed in the literature (for reviews see [12,13]). The most recent bloat theory, Crossover Bias [9,2,4,11], explains code growth in tree-based GP by the effect that standard subtree crossover has on the distribution of program lengths in the population. Developed alongside Crossover Bias, Operator Equalisation (OpEq) is the newest bloat control method available [3,14,15]. Although still in its infancy, it has already proven to be very successful in both benchmark and real life problems [15,16].

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 272–281, 2010.

Classification Trees are a non-parametric, non-linear rule based classifier that generates classification rules through an induction procedure described in [1]. They are based on a hierarchical decision scheme where the feature space is subject to a binary recursive partitioning that successively splits the data. This is the basic divide and conquer methodology used in the C4.5 algorithm, with some differences in terms of the tree structure, the splitting criteria, the pruning method, and the way missing values are handled [5]. In the Classification and Regression Tree (CART) algorithm [1], heuristics techniques are used to achieve an inverted tree type structure, starting in a root node with all the data, and generating descendent nodes with a series of splitting decisions (if-then rules) until terminals are reached, meaning that the classes are all differentiated. CART is a popular and successful classification algorithm.

This paper compares GP and CART on a problem of identification of burned areas in satellite imagery. Additionally, it studies how the OpEq bloat control technique affects the quality of the solutions provided by GP. The goal is to understand if and how a GP approach armed with the latest bloat control technique can beat the high quality solutions provided by a well established method like CART. The merit of each approach is assessed not only by its classification accuracy, but also by the ability to predict the correctness of its own classifications, and the ability to provide solutions that are human readable and robust to data inaccuracies. We do not consider the time it takes to deliver a solution with each of the techniques, since that is not such an important factor in this type of application.

The next section briefly describes OpEq, since it is still a very recent and widely unknown bloat control technique. Section 3 provides details regarding the data, while Section 4 describes the techniques and parameters used for the experiments. Section 5 reports and discusses the results, and Section 6 concludes and suggests future developments of this work.

2 Operator Equalisation

Developed alongside the Crossover Bias theory [9,2,4,11], OpEq is one of the few bloat control methods based on a precise theoretical study. By filtering which individuals are accepted in each new generation, this technique allows accurate control of the distribution of program lengths inside the population, easily biasing the search towards smaller or larger programs. In the first published version of OpEq [3] the user had to specify the desired length distribution, called target, as well as a maximum allowed program length. Both remained static throughout the entire run. Each newly created individual was accepted or rejected for the new generation based on its length and the target distribution.

This first implementation was shortly followed by a dynamic version of OpEq, called DynOpEq [14], where both the target and the maximum allowed program length are self adapted along the run. In DynOpEq the acceptance or rejection of the newly created individuals is based not only on their length, but also on their fitness. In [14] DynOpEq was tested on four well known GP benchmark

problems (symbolic regression, artificial ant, even-parity and multiplexer), while in [15] DynOpEq was used on a real life drug discovery application (prediction of human oral bioavailability).

Also in [15] a new version of OpEq was defined. Contrarily to DynOpEq, it does not reject any individuals, and instead transforms them by slightly mutating their genotype until they reach the desired length. This new implementation has been called MutOpEq. Both DynOpEq and MutOpEq were very successful in controlling bloat without harming fitness. In [16] both these techniques were used in another drug discovery application (prediction of toxicity), once again with much success.

3 Data

The satellite image used in this study is an orthorectified Landsat 7 ETM+ scene over Guinea-Bissau, corresponding to Path/Row 204/52, and downloaded free of charge from the USGS site[1]. The image was collected in the end of the dry-season (May 13, 2002), thus ensuring the presence of burned areas and the highest possible discrimination among vegetation types in a forested/wooded savanna tropical ecosystem. Landsat 7 ETM+ images consist of eight different bands (of which we used the first seven) with a radiometric resolution of 8 bits. Thus, the reflectance of objects on the surface is encoded on a 30 meter resolution grid as a digital number (DN) varying between 0 and 255, for each spectral band.

Visual inspection of the RGB combination of bands 7, 4 and 3, allows depicting burned areas very clearly [8]. Using a 7-4-3 RGB combination as a basis, samples of burned and not burned areas were delimited on screen. The pixels included in those polygons were extracted to constitute the data for our study, in a total of 3637 samples (1889 burned and 1748 not burned). Each sample consists of the observed DN values for the seven variables (the seven bands) and the corresponding target value, burned (1) or not burned (0).

To assess the learning and generalization ability of the different classification techniques, 30 random partitions were created from this data set, each containing 70% of the samples for training, and 30% for testing. Neither GP nor CART require any normalization or other pre-processing of the data. However, since GP is used in a regression-like fashion (see Sect. 5) the input variables were scaled from $[0, 255]$ to $[0, 1]$ so they would be closer to the expected output values (0 or 1).

4 Methods

Four different techniques were studied. One of them is the well established CART algorithm, used as a single tree classifier with linear combination of the variables, the Gini index criteria for node splitting, equal class prior probabilities, equal classification error costs for the two classes, and terminal nodes with a minimum

[1] http://glovis.usgs.gov/

of 20 observations. Due to its high popularity and successfulness, CART is here regarded as the baseline.

The others are GP techniques. The first is standard GP (StdGP), the most traditional implementation of tree-based GP, using minimal bloat control implemented as the historical maximum tree depth of 17 [6]. The others are the two flavors of OpEq (see Sect. 2), DynOpEq and MutOpEq, none of them using any depth or size limits. We used populations of 500 individuals allowed to evolve for 200 generations, a function set containing only the four binary operators $+$, $-$, $\times$, and $/$, protected as in [6], and a terminal set containing only the seven variables. We adopted the remaining parameters from [15,16].

To perform the experiments we used CART 5.0[2] and a modified version of GPLAB 3.0[3]. Each of the four techniques was run 30 times, once in each partition of the data set. Statistical significance of the null hypothesis of no difference was determined with pairwise Kruskal-Wallis non-parametric ANOVAs at $p = 0.05$.

5 Results and Discussion

A solution provided by CART is a set of if-then rules that for each sample indicates which class it belongs to. It is possible to use GP to evolve classification rules, but in this work we used it as if we were dealing with a symbolic regression problem. Therefore, a GP solution is a single function that for each sample returns a real value, expected to be close to either 0 or 1. This value is then converted to its closest class. In the following sections we add more details on this subject, while reporting and discussing the results from four different points of view: accuracy, reliability, complexity and robustness.

5.1 Accuracy

Accuracy is the first, sometimes the only, indicator of classifier quality. All four techniques achieved very high accuracy values (calculated as the percentage of correctly classified samples) in both training and test sets. Figure 1 shows a boxplot containing these results.

None of the techniques revealed a significant difference in accuracy between the training set and the test set. This means there is no overfitting, a fact that would allow us to report the remaining results in the unpartitioned original data set. However, a closer look reveals that the relative merit of the different techniques is not the same in both sets. While in the training set a number of techniques revealed significant differences between each other, in the test set there was only one significant difference: DynOpEq achieved better accuracy than CART. Therefore, we report the remaining results only in the test set (more precisely, in the 30 test sets).

Given that CART and GP operate differently from each other, and return solutions in distinct formats (rules vs functions), it is interesting to check whether

[2] http://salford-systems.com/cart.php
[3] http://gplab.sourceforge.net/

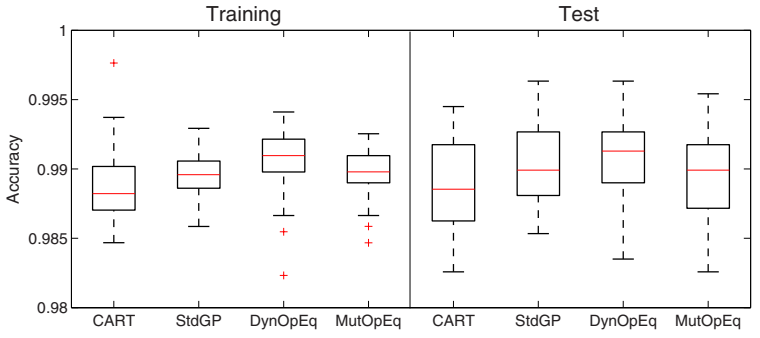


Fig. 1. Boxplot of the accuracy values on the training and test sets. MutOpEq has outliers at 95.5% (training) and 94.5% (test), not shown.

the few misclassified samples are the same in both approaches. The answer is that almost half of the total misclassifications is common to CART and GP. We concluded this by measuring the overlap between the sets of misclassified samples (as if they were independent across the 30 test sets) of each technique. The percentage of common samples to CART and each of the GP techniques StdGP, DynOpEq and MutOpEq is 48%, 45% and 44%, respectively. The overlap among the three GP techniques ranges from 56% to 65%. In a set where the percentage of misclassifications is so low, the probability of achieving such a large overlapping by pure chance is extremely low.

5.2 Reliability

Along with the classification of each sample, CART also returns learn-based fractions in each terminal node as predicted probabilities that the samples actually belong to each class. This is very important to the end user, as long as this probability faithfully represents the reliability of the classification. When using GP as a regression tool like we did in this work, it is also possible to calculate the probability of correctness, or reliability. For each classified sample i the reliability value r_i is calculated as $1 - d_i$, where d_i is the distance between the real value returned by the GP function and its nearest class. Because the GP function can potentially return any real value, we apply *if* $d_i > 0.5$ *then* $d_i = 0.5$ so that the reliability values are always in the range between 0.5 and 1.

Figure 2(a) shows the distribution of the reliability values for the correctly and incorrectly classified samples (once again as if they were independent across the 30 test sets) for all techniques. It is immediately apparent that CART concentrates its reliability values in a very narrow interval (any value below 0.95 is an outlier), while the GP techniques exhibit a high dispersion of values. For GP, the incorrectly classified samples use the entire range between 0.5 and 1, while the correct classifications use narrower intervals and consider a tremendous amount

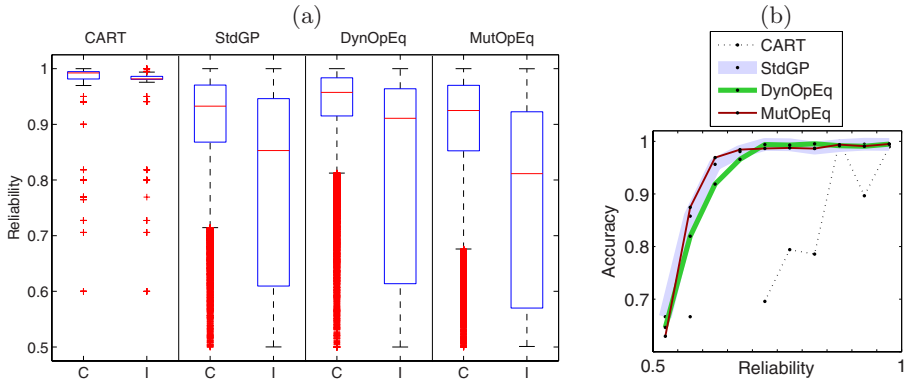


Fig. 2. (a) Distribution of reliability values for correctly (C) and incorrectly (I) classified samples; (b) Accuracy obtained for samples in different reliability intervals

of outliers[4]. In all cases the overlapping of boxes and whiskers between correct and incorrect classifications is very high, particularly in the CART technique, making it difficult to predict the correctness of the classifications.

Nevertheless, the reliability does have some predictive value. Figure 2(b) plots the accuracy obtained in all test samples, for different reliability intervals. Ten intervals were considered using equidistant points between 0.5 and 1. Given that CART uses few different reliability values, we could not use a higher number of intervals or most of them would be left empty in CART (with 10 intervals, already three of them are empty). This plot shows that with the GP techniques it is possible to use the reliability value to predict the accuracy, but not so much with CART. DynOpEq is the one providing a finer prediction.

5.3 Complexity

It is feasible to compare different solutions obtained with different CART experiments in terms of the number and length of their rules. It is also possible to compare the length of the different solutions returned by GP, preferably taking care to simplify the raw expressions provided (although many times only the expressions containing redundant parts benefit from this procedure). We can even use these measurements to talk about the complexity of the solutions, using the term complexity in a very informal manner, as a mere indicator of whether the solution is easily readable and understood by a human user. However, comparing the complexity of a set of rules to the complexity of a single function is an impossible task. Figure 3 shows examples of different solutions provided by the four techniques.

The CART solution shown is one of the shortest, while the GP solutions were the result of some simplification. The solution shown for MutOpEq is surprisingly

[4] The "bars" below the lowest whiskers are just a high concentration of crosses representing the outliers.

Example of solution by CART:

if $(0.641x_4 - 0.768x_6 <= -0.357192)$ **and** $(x_2 <= 0.335)$ **then** $class = 1$
if $(0.641x_4 - 0.768x_6 <= -0.357192)$ **and** $(x_2 > 0.335)$ **then** $class = 0$
if $(0.641x_4 - 0.768x_6 > -0.357192)$ **then** $class = 0$

Example of solution by StdGP:

$$y = x_6 - (x_2 + x_4) + (x_6 + x_7 - 3x_4)\Big(3x_6 - (x_1 + 3x_4) + x_1\big(2x_1 + x_6 + x_7 - (x_3 + 5x_4) + (7x_6 - 10x_1 - 4x_4 + 12x_7 - 2x_2 - 7x_3)(8x_1 - 3x_6 + x_4 - 3x_7)\big)\Big)$$

Example of solution by DynOpEq:

$$y = \frac{x_3 x_6}{x_7} + x_1 - x_4 + x_6^2 - \frac{x_6}{x_5} + \frac{x_5 x_6^3}{x_1 x_3^2}\left(\frac{1}{x_3} - \frac{x_7}{x_5}\right)\left(\frac{x_5}{x_3} + \frac{1 - \frac{x_3 x_5}{x_6^2 x_7}}{\frac{x_5^2}{x_7^2} + x_6 + x_7 - x_3 - \frac{x_6}{x_5}}\right)$$

Example of solution by MutOpEq:

$$y = x_2 - x_5 + x_3 x_6 + \frac{x_3}{x_6}$$

(rule to apply to all GP solutions: if $y < 0.5$ **then** $class = 0$ **else** $class = 1$**)**

Fig. 3. Examples of solutions provided by the four techniques

shorter than its original raw expression, a truly uncommon feat that this technique was able to achieve in several different runs. StdGP produced functions that, besides exhibiting a different format than the solutions by the OpEq techniques (see Sect. 5.4), are generally much longer, due to bloat. Figure 4(a) shows the evolution of the average length of the solutions along the generations (median of 30 runs). While StdGP keeps increasing solution length, both DynOpEq and MutOpEq stabilize it very soon and maintain it along the entire evolution. Figure 4(b) shows the relationship between the average length and the best fitness achieved in the test set (median of 30 runs), along the evolution (there is an implicit downwards timeline along the fitness axis). Here we can see the turning point when the OpEq techniques keep improving fitness while maintaining the average length of the solutions. Unlike StdGP, both DynOpEq and MutOpEq are completely bloat free.

5.4 Robustness

Looking at the solutions provided by CART, we noticed that the same input variables, x_4 and x_6, are always selected as the most important in the classification, while others are always considered little important. The same order of importance was found in all 30 solutions (x_4, x_6, x_2, x_5, x_3, x_1, x_7), suggesting a large degree of homogeneity among the different experiments. GP is known for finding extremely varied solutions from one run to the other, so we decided to check if,

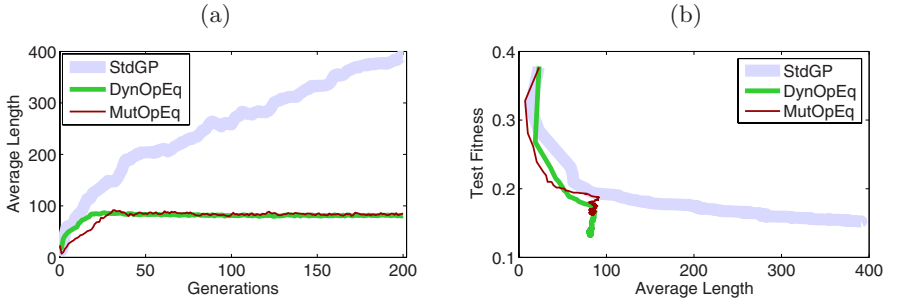


Fig. 4. (a) Evolution of the average length of the solutions along the generations; (b) Relationship between average length and test fitness along the evolution

along the 30 runs, GP had some preference for x_4 or x_6, or indeed any other variable. Without doing any sensitivity analysis, we simply counted the median number of times that each variable appears in each of the 30 solutions. This can hardly be regarded as an indicator of variable importance, but at least it can reveal the differences among the techniques. Besides the seven variables, we also counted the four arithmetic operators in the same way. The results did reveal something interesting. In all three GP techniques the least repeated variables were x_1, x_2 and x_3, by no particular order. The most repeated were x_4 and x_6 in StdGP and MutOpEq, and x_5 and x_7 in DynOpEq. Although StdGP always used both x_4 and x_6 in each of the 30 solutions, both OpEq techniques could find good solutions without using one or the other. In terms of the operators, the most interesting finding was that StdGP uses $+$ and $-$ almost exclusively, a few $\times$, and absolutely no $/$, while the OpEq techniques use all operators. It is beyond the scope of this work to analyze the reasons behind these differences.

Given the varied characteristics of the solutions obtained by the different techniques, we performed a simple test to check which techniques produce more robust solutions, where robustness is measured as the ability to deal with noisy data. It is expected that a technique that always depends on the same variables, like CART,

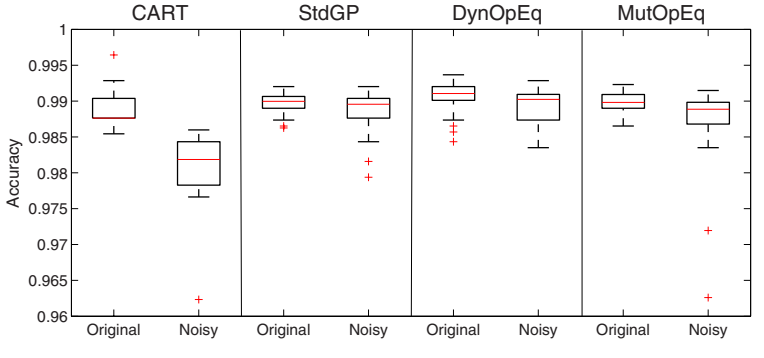


Fig. 5. Boxplot of the accuracy values on the original and noisy data sets. MutOpEq has additional outliers at 95.2% (original) and 92.9% (noisy), not shown.

is less robust to perturbations of those same variables. So we used a new data set that is exactly the same as the original unpartitioned data set except for the fact that in each sample the variable x_4 has its value randomly increased or decreased by 10% ($x_4 = x_4 \pm 0.1 \times x_4$). Figure 5 shows the accuracy obtained in this noisy set, compared to the original set. It is not surprising to know that the results of CART suffered a very significant decrease of accuracy. What is interesting is that, among the GP techniques, both OpEq also suffered significant differences, while StdGP did not - precisely the only GP technique where all the solutions used x_4! Recalling previous work [15] where StdGP exhibited a lot more "resistance" to overfitting than DynOpEq, and considering that failure to cope with noisy data is indeed a suggestion of overfitting, we wonder, once again, about the possible "protective" (and counterintuitive) effect that bloat may have on overfitting.

6 Conclusions and Future Work

We have compared GP and CART on a problem of identification of burned areas in satellite imagery. Two of the GP techniques used the latest bloat control method available, OpEq. The quality of the solutions was assessed in terms of accuracy, reliability, complexity and robustness, with the goal of understanding whether GP can outperform the popular and successful CART, and how does OpEq contribute to it. The first OpEq technique was able to achieve significantly higher accuracy than all the other techniques. All the GP techniques showed good predictive ability of the correctness of their own classifications, while CART was not very reliable. In terms of complexity, the second OpEq technique was the only one providing surprisingly short solutions that challenge the simplest rules produced by CART. Another surprise was the robustness that standard GP exhibited to noisy data. Given that one of the OpEq techniques equalled or surpassed CART in all the tests, our main conclusion is that using GP provides advantages even in a problem where a well established and successful method like CART already produces excellent results. Among all the experiments, GP was in fact able to find models that are more reliable, and as accurate, robust and human readable as the ones from CART.

In the future we plan to use GP and CART on harder classification tasks where a comparison can be made also in terms of overfitting and the potential of each technique to avoid it. In a more general way, we want to study what makes a particular technique more or less prone to overfitting, and how that may be related (or unrelated) to bloat. We also plan to use GP to evolve classification rules and compare the results with these. Finally, we want to produce burned area maps using the best technique.

Acknowledgements

The authors acknowledge projects "CARBOVEG-GB – Quantifying the carbon stocks and sink effects in the forests of Guinea-Bissau" (APA/RELAC 2006) and "EnviGP – Improving Genetic Programming for the Environment and Other Applications" (PTDC/EIA-CCO/103363/2008).

References

1. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees, Wadsworth (1984)
2. Dignum, S., Poli, R.: Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In: Thierens, D., et al. (eds.) Proceedings of GECCO 2007, pp. 1588–1595. ACM Press, New York (2007)
3. Dignum, S., Poli, R.: Operator equalisation and bloat free GP. 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. 110–121. Springer, Heidelberg (2008)
4. Dignum, S., Poli, R.: Crossover, sampling, bloat and the harmful effects of size limits. 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. 158–169. Springer, Heidelberg (2008)
5. Kohavi, R., Quinlan, J.R.: Decision-tree discovery. In: Klosgen, W., Zytkow, J.M. (eds.) Handbook of Data Mining and Knowledge Discovery, ch. 16.1.3, pp. 267–276. Oxford University Press, Oxford (2002)
6. Koza, J.R.: Genetic programming – on the programming of computers by means of natural selection. MIT Press, Cambridge (1992)
7. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2002)
8. Pereira, J.M.C., Sá, A.C.L., Sousa, A.M.O., Silva, J.M.N., Santos, T.N., Carreiras, J.M.B.: Spectral characterisation and discrimination of burnt areas. In: Chuvieco, E. (ed.) Remote Sensing of Large Wildfires in the European Mediterranean Basin, pp. 123–138. Springer, Heidelberg (1999)
9. Poli, R., Langdon, W.B., Dignum, S.: On the limiting distribution of program sizes in tree-based genetic programming. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 193–204. Springer, Heidelberg (2007)
10. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming, http://www.gp-field-guide.org.uk (2008) (With contributions by J. R. Koza), http://lulu.com
11. Poli, R., McPhee, N.F., Vanneschi, L.: The impact of population size on code growth in GP: analysis and empirical validation. In: Keijzer, M., et al. (eds.) Proceedings of GECCO 2008, pp. 1275–1282. ACM Press, New York (2008)
12. Silva, S.: Controlling bloat: individual and population based approaches in genetic programming. PhD thesis, Dep. Informatics Engineering, Univ. Coimbra (2008)
13. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genet. Program. Evolvable Mach. 10(2), 141–179 (2009)
14. Silva, S., Dignum, S.: Extending operator equalisation: Fitness based self adaptive length distribution for bloat free GP. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 159–170. Springer, Heidelberg (2009)
15. Silva, S., Vanneschi, L.: Operator equalisation, bloat and overfitting - a study on human oral bioavailability prediction. In: Rothlauf, F., et al. (eds.) Proceedings of GECCO 2009, pp. 1115–1122. ACM Press, New York (2009)
16. Vanneschi, L., Silva, S.: Using operator equalisation for prediction of drug toxicity with genetic programming. In: Lopes, L.S., Lau, N., Mariano, P., Rocha, L.M., et al. (eds.) EPIA 2009. LNCS, vol. 5816, pp. 65–76. Springer, Heidelberg (2009)

Genetic Algorithms for Training Data and Polynomial Optimization in Colorimetric Characterization of Scanners

Leonardo Vanneschi, Mauro Castelli, Simone Bianco, and Raimondo Schettini

Department of Informatics, Systems and Communication (D.I.S.Co.)
University of Milano-Bicocca, Milan, Italy
{vanneschi,castelli,bianco,schettini}@disco.unimib.it

Abstract. Generalization is an important issue in colorimetric characterization of devices. We propose a framework based on Genetic Algorithms to select training samples from large datasets. Even though the framework is general, and can be used in principle for any dataset, we use two well known datasets as case studies: training samples are selected from the Macbeth ColorCheckerDC dataset and the trained models are tested on the Kodak Q60 photographic standard dataset. The presented experimental results show that the proposed framework has better, or at least comparable, performances than a set of other computational methods defined so far for the same goal (Hardeberg, Cheung, CIC and Schettini). Even more importantly, the proposed framework has the ability to optimize the training samples and the characterizing polynomial's coefficients at the same time.

1 Introduction

Many devices, like scanners or printers, have their own reference systems for the specification of color (device-dependent spaces). To facilitate the reproduction of colors on various devices and supports, it is often useful to employ a system of description that allows us to define the color in a univocal fashion, i.e. in a device-independent space, separating the way colors are defined from the way the various devices represent them. A point in RGB space indicates how a color stimulus is produced by a given device, while a point in a colorimetric space, such as CIELAB space, indicates how the color is perceived in standard viewing conditions. Now let us consider the function that at every point in the device dependent space associates the colorimetric value of the corresponding color. The colorimetric characterization of a scanner device means to render this function explicitly. It must take into account the peculiar characteristics of the device; consequently, every device calls for specific conversion functions. Several different approaches to the characterization problem have appeared so far (see for instance [5,14,4]).

High-order multidimensional polynomials are often used for scanner characterization. However, since there is no clear relationship between polynomials adopted and imaging device characteristics, they must be empirically determined and defined for each device, and for a specific device, each time a change is made in any component of the system [11]. Using an average color error as the functional to be minimized,

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 282–291, 2010.

the polynomial coefficients can be found by using the Least Square method or the Total Least Squares method [10]. More powerful methods, such as Total Color Difference Minimization and CIELAB Least Squares minimization, have been recently proposed [13] to minimize non linear functionals that take into account the average CIELAB colorimetric error. An evolutionary framework integrating Genetic Programming and Genetic Algorithms (GAs) [9,7] was proposed in [1,2] for optimizing the polynomial's shape and coefficients.

All the results presented in the contributions quoted above confirm that the accuracy of the characterization increases as the number of terms in the polynomial increases and also that there is not a simple way to avoid data over-fitting. For this reason, data overfitting avoidance and generalization is becoming more an more an issue for colorimetric characterization of scanners. In this work, we address this issue presenting a GA based system to find, out of large datasets, those training samples that allow us a reasonable generalization. GAs performance is compared here with some of the most well known state of the art methods for training samples selection. Besides showing that GAs have better, or at least comparable, performance than those methods in selecting training samples, the goal of this work is also to show that GAs can, at the same time, choose accurate polynomial coefficients, an ability that the other methods considered here do not have. The datasets used to evaluate the color rendition accuracy and the generalization ability of the proposed color selection methods are two widely used datasets: the Macbeth ColorCheckerDC (MDC) and the Kodak Q60 photographic standard (IT8). The interested reader is referred for instance to [2] for an introduction to these datasets.

This paper is structured as follows: Section 2 briefly introduces the scanner characterization problem. Section 3 contains an introduction to four state of the art methods for training samples selection: Hardeberg, Cheung, CIC and Schettini. In Section 4 we present our GA based system. In Section 5 we report and discuss the obtained experimental results. Finally, Section 6 concludes the paper and proposes ideas for future research.

2 Scanner Characterization

The basic model that describes the response $\rho = [R,G,B]$ of a three-channel color scanner can be formulated as [13,14]: $\rho = \mathcal{F}(\mathbf{SIR} + \mathbf{n})$, where ρ is a 3×1 vector, $\mathbf{S}$ is the $3 \times N$ matrix formed by stacking the scanner spectral sensitivities row-wise, $\mathbf{I}$ is the $N \times N$ diagonal matrix whose elements are the samples of the scanner-illuminant spectral power distribution, $\mathbf{R}$ is the $N \times 1$ vector of the reflectance of the surface being scanned, $\mathbf{n}$ is the 3×1 noise vector and $\mathcal{F}$ is an optoelectronic conversion function representing the input-output nonlinearity that may characterize the scanner response. Similarly, the CIEXYZ tristimulus values, denoted by a 3×1 vector $\mathbf{s}$, can be defined as: $\mathbf{s} = \mathbf{CLR}$, where $\mathbf{C}$ is the $3 \times N$ matrix of the CIEXYZ color matching functions and $\mathbf{L}$ is the $N \times N$ diagonal matrix whose elements are the samples of the viewing-illuminant spectral power distribution.

The characterization problem is to find the mapping $\mathcal{M}$ which transforms the recorded values ρ to their corresponding CIEXYZ values $\mathbf{s}$: $\mathbf{s} = \mathcal{M}(\rho)$. Usually, given an optoelectronic conversion function $\mathcal{F}$, a m^{th}-order polynomial mapping $\mathbf{M}$ is applied

to the linearized data $\mathcal{F}^{-1}(\rho)$ to obtain $\mathbf{s}$. The general m^{th}-order polynomial $P(R,G,B)$ with three variables can be given as: $P(R,G,B) = \sum_{i=0}^{m}\sum_{j=0}^{m}\sum_{k=0}^{m} R^i G^j B^k$, with $i+j+k \leq m$. Given the scanner response ρ, their linearized values $\mathcal{F}^{-1}(\rho)$ and the polynomial model P to use, we can calculate the polynomial expansion $\mathbf{r}$ of $\mathcal{F}^{-1}(\rho)$ as $\mathbf{r} = P\left(\mathcal{F}^{-1}(\rho)\right)$. Using the polynomial modeling, the previous equation then becomes: $\mathbf{s} = \mathbf{M}P\left(\mathcal{F}^{-1}(\rho)\right) = \mathbf{Mr}$. The first step to find the matrix $\mathbf{M}$ is to select a collection of color patches that spans the device gamut. The reflectance spectra of these N_c color patches will be denoted by $\mathbf{R}_k$ for $k \in \{1,\ldots,N_c\}$. These patches are measured using a spectrophotometer or a colorimeter which provides the device independent values: $\mathbf{s}_k = \mathbf{CLR}_k \quad \text{with} \quad k \in \{1,\ldots,N_c\}$. Without loss of generality, $\mathbf{s}_k$ can be transformed in any colorimetric or device independent values. The same N_c patches are also acquired with the scanner to be characterized providing $\rho_k = \mathcal{F}\left(\mathbf{SIR}_k + \mathbf{n}\right)$ with calculated polynomial expansions $\mathbf{r}_k$, for $k \in \{1,\ldots,N_c\}$. In equation, the characterization problem is to find the matrix $\mathbf{M}$:

$$\mathbf{M} = \arg\left(\min_{M\in\mathbb{R}^{3\times q}} \sum_{k=1}^{N_c} ||M\mathbf{r}_k - \mathcal{L}(\mathbf{s}_k)||^2\right) \tag{1}$$

where $\mathcal{L}(\cdot)$ is the transformation from CIEXYZ to the appropriate standard color space chosen and $||\cdot||$ is the error metric in the color space. $\mathbf{M}$ is a $3\times q$ matrix, where q is the number of terms of the polynomial $P(R,G,B)$; the number of terms q is related to the order m of the polynomial by: $q = \left[\sum_{k=1}^{m}\binom{k+2}{2} + 1\right]$. Being $\mathbf{M}$ a $3\times q$ matrix, the problem of finding $\mathbf{M}$ amounts to determine $3q$ coefficients; to have enough equations to solve for the $3q$ unknowns and to deal with a less ill-posed problem, we have to use $N_c \geq q$ different color patches. In other words, the bigger is the order of the polynomial, the greater is the number of its terms and consequently the greater is the number of different color patches we have to use. Different functionals to be minimized can be defined to find the unknown matrix $\mathbf{M}$. In this work, we consider the ΔE_{94} error (see for instance [13]). The use of higher order polynomials as characterization functions permits to reach higher colorimetric accuracy on the training set at the expense of generalization capability. In order to have a single characterization function for all the possible input supports that can be acquired with the scanner, and to not derive a new characterization function for each different input support, it is thus necessary to design new optimization strategies which permit to find characterization functions with higher generalization capability. In the next section, we briefly introduce some of the most known state of the art methods to accomplish this task. The generalization ability of those methods will be compared with the one of the proposed GA system, introduced in Section 4.

3 State of the Art Methods for Training Samples Selection

Hardeberg. The method proposed by Hardeberg *et. al* [8] selects a set of reflectance samples from a large set of patches so that the chosen spectra in reflectance space are as different as possible. It is an iterative procedure. The first sample selected r_1 is the spectral reflectance function with the largest root mean square value: $RMS(r_{i_1}) \geq \mathrm{RMS}(r_k) \forall k = 1,\ldots,K$ where K is the number of samples in the considered reflectance dataset, and

$\mathrm{RMS}(r) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i}$ where n is the number of samples of the reflectance spectra. The second sample selected r_2 is the one which minimizes the condition number of the matrix $[r_1 r_2]$. Let us denote λ_{max} and λ_{min} respectively the largest and the smallest singular values of the matrix $[r_1 r_2]$, then the second sample selected is the one which satisfies: $\frac{\lambda_{max}[r_{i_1} r_{i_2}]}{\lambda_{min}[r_{i_1} r_{i_2}]} \leq \frac{\lambda_{max}[r_{i_1} r_k]}{\lambda_{min}[r_{i_1} r_k]} \quad \forall k = 1,\ldots,K, k \neq i_1$. The j-th sample is then selected in a similar way: $\frac{\lambda_{max}[r_{i_1}\ldots r_{i_j}]}{\lambda_{min}[r_{i_1}\ldots r_{i_j}]} \leq \frac{\lambda_{max}[r_{i_1}\ldots r_{i_{j-1}} r_k]}{\lambda_{min}[r_{i_1}\ldots r_{i_{j-1}} r_k]} \quad \forall k = 1,\ldots,K, k \neq i_1,\ldots,i_{j-1}$.

Cheung. The method proposed by Cheung and Westland [3] also selects the samples which are as different as possible. It is similar to the method proposed by Hardeberg *et. al* but it differs by the way the sample similarity is computed. The first sample selected is the one with the lowest variance of spectral reflectance. The second sample selected is the one with the largest ΔE colorimetric difference from the first sample. The j-th sample is selected in a similar way: it is first calculated the ΔE colorimetric difference from all the $j-1$ selected samples, and then its minimum values are recorded. It is then selected as the j-th sample, the one having the maximum minimum ΔE distance from the samples already selected.

CIC. The method proposed by Chou *et. al* [6] makes use of two different data sets to select the best training samples: the source data and the decision data sets. Assume that there are N samples from the source data set. Each of them is considered as candidate and used to train the characterization model. Each model is then applied to the decision data set. The sample which returns the model with the lowest ΔE colorimetric error on the decision set is selected as the first sample. The procedure is then iterated to identify the sample that paired with the first one gives the model with the lowest error on the decision set. The subsequent samples are selected in the same way.

Schettini. The method proposed by Schettini *et. al* [12] uses the values recorded by the acquisition device to select the samples that are as different as possible from each other. The first sample selected is the one satisfying: $\mathrm{SSE}(r_{i_1}) \geq \mathrm{SSE}(r_k) \quad \forall k = 1,\ldots,K$ where $\mathrm{SSE}(r) = \sqrt{\sum_{i=1}^{n} {x_i}^2}$. The maximum absolute difference (MAD) in terms of camera coordinates for the recorded samples is calculated from the first selected sample. The one having the maximum MAD value is then selected. The other samples are similarly selected: the j-th sample is the one having the maximum minimum MAD value from all the $j-1$ samples already selected.

4 The Proposed GA System

In this work, we have used GAs for three different tasks: (i) optimizing training samples to improve generalization using fixed polynomial coefficients (for this first task, GAs performance is compared to the one of the methods presented in Section 3); (ii) optimizing polynomial coefficients using fixed training samples and (iii) optimizing polynomial coefficients and training samples at the same time. The ways GAs have been used for tasks (i), (ii) and (iii) are described below.

(i) GAs for optimizing training samples using fixed polynomial coefficients. Let $D = (d_{ij})$ be an $N \times M$ matrix representing the MDC dataset, where each line represents

one of the samples and each column a feature. Our goal is to select a number of training samples smaller or equal to a given $k < N$. A GA individual is represented by a k-dimensional vector $\mathbf{v} = (v_1, v_2, ..., v_k)$ where, for each $i = 1, 2, ..., k$, v_i is an integer number between 1 and N. Intuitively, each GA individual represents a set of samples taken from the MDC dataset.

The fitness of each GA individual $\mathbf{v}$ is calculated as follows: we use only the samples represented by individual $\mathbf{v}$ for finding the unknown matrix $\mathbf{M}$ in Equation (1). Given that fitness is the most time-consuming task in an evolutionary algorithm, and it has to be calculated many times during the evolution, we use a rather simple and fast method to find matrix $\mathbf{M}$: we minimize functional $\mathcal{H}_{LS} = ||\mathbf{s} - \hat{\mathbf{s}}||_2$ with $\hat{\mathbf{s}} = \mathbf{Mr}$ using the Least Squares minimization [10]. Analytically, the matrix $\mathbf{M}$ can be easily found by: $\mathbf{M} = \mathbf{sr}^T(\mathbf{rr}^T)^{-1}$. Successively, we consider as fitness value of individual $\mathbf{v}$ the ΔE_{94} error obtained by the trained model on the $N - k$ samples included in the MDC dataset and not represented by $\mathbf{v}$.

The genetic operators we use work as follows: mutation replaces, with a given probability p_m, each allele in an individual with a random integer number chosen with uniform distribution between 1 and N (it intuitively corresponds to the replacement of a sample of the MDC dataset with another random one). Crossover is similar to the standard one-point GAs crossover [9,7], except that we have explicitly avoided the possibility of a number to appear in more than one allele in the offspring.

(ii) GAs for optimizing polynomial coefficients using fixed training samples. For optimizing the polynomial coefficients, we use binary coded GAs. Each allele corresponds to one term of the complete polynomial of a given degree. The value 0 indicates absence of that term in the polynomial represented by the individual, while the value 1 indicates presence of the term. For a given $k < N$, k samples in the MDC dataset are selected at random at each GA run, and fitness is calculated as follows: we use only these k samples and the polynomial represented by the GA individual for finding the unknown matrix $\mathbf{M}$ in Equation (1) with Least Squares minimization. Successively, we consider as fitness value the ΔE_{94} error obtained by the trained model on the other $N - k$ samples included the MDC dataset. In this case, genetic operators are standard binary coded GAs crossover and mutation [9,7].

(iii) GAs for optimizing polynomial coefficients and training samples at the same time. When the target is to optimize polynomial coefficients and training samples at the same time, GA individuals are represented using diploid chromosomes [7], i.e. they are represented by two strings. The first one of these strings is coded as the individuals described in (i) and it has exactly the same meaning, while the second one is coded as the individuals described in (ii). In other words, the first string represents the selected training samples and the second one the chosen polynomial coefficients.

Fitness of such an individual is calculated as follows: we use the samples represented in the first string and the polynomial represented in the second string for finding the unknown matrix $\mathbf{M}$ in Equation (1) using Least Squares minimization. Successively, we consider as fitness value the ΔE_{94} error obtained by the trained model on the $N - k$ samples included in the MDC dataset and not represented by the first string, obtained using the polynomial represented by the second string.

The genetic operators this time work as follows: given two individuals **v** and **w**, the first string of the offspring is given by the offspring of the crossover between the first strings of **v** and **w** as described in (i) and the second string of the offspring is given by the offspring of the crossover between the second strings of **v** and **w** as described in (ii). Mutation is applied with probability p_m to all alleles of both strings, but for the first string the used mutation is the one described in (i), while for the second string mutation is the one described in (ii).

5 Experimental Results

In this section we outline the results produced by the exploited computational methods. Results are organized as follows: we first show the results obtained by GAs and the other considered methods keeping fixed polynomials and looking for the training samples that allow the best generalization. Secondly, we present the results produced by GAs using fixed training samples and looking for the best polynomial's coefficients. Finally, we present the results obtained by GAs where the learning process involves both the training samples and the polynomial's coefficients estimation at the same time. In all cases, the GA parameters we have used are: population size equal to 100 individuals; number of generations equal to 150; crossover probability p_c equal to 0.95; mutation probability p_m equal to 0.1; tournament selection of size 3; elitism (i.e. copy of the best individual unchanged in the next population at each generation). In all the experiments described in this section, models have been trained on selected subsets of the MDC dataset, while the reported results have been obtained by testing the models on the IT8 dataset, that has absolutely not been used during the training phase.

Optimizing training samples using fixed polynomial coefficients. In this first set of experiments, we use four different fixed polynomials and the goal is to find the training samples, extracted from the MDC dataset, that allow the best generalization ability. We performed several experiments in order to extract 12, 18, 24, 50, 75 and 100 samples

Table 1. Values of the ΔE_{94} error on the test set (IT8) obtained by the models trained on the samples selected by each studied method. Fixed complete first degree polynomial without offset. The minimum, maximum, mean, median, 90 percentile and standard deviation of ΔE_{94} error are reported. For GAs results averaged over 50 independent runs are reported.

	n. samp.	Min	Max	Mean	Median	90 prc	std dev		n. samp.	Min	Max	Mean	Median	90 prc	std dev
HARD	12	0.91	13.94	4.87	4.61	7.92	2.41	HARD	50	0.58	11.01	4.15	3.99	6.33	1.89
CHEU	12	0.56	10.27	4.16	4.23	6.37	1.98	CHEU	50	0.54	9.32	3.96	3.69	6.01	1.71
CIC	12	0.70	9.92	3.79	3.56	6.35	2.04	CIC	50	0.69	10.31	3.89	3.69	6.21	1.98
SCHE	12	0.66	11.32	4.52	4.49	7.81	2.63	SCHE	50	0.39	12.13	4.36	4.22	7.43	2.58
GAs	12	0.52	9.33	3.95	3.79	6.40	3.77	GAs	50	0.63	9.85	3.84	3.58	6.35	4.03
HARD	18	0.80	12.69	4.68	4.52	8.05	2.23	HARD	75	0.41	10.43	4.53	4.28	7.74	2.18
CHEU	18	0.61	10.96	4.26	4.07	6.70	1.99	CHEU	75	0.89	10.14	4.02	3.76	6.16	1.78
CIC	18	0.72	9.74	3.81	3.53	6.09	1.92	CIC	75	0.69	10.04	3.86	3.65	6.12	1.94
SCHE	18	0.10	9.46	4.20	4.10	7.38	2.44	SCHE	75	0.25	11.32	4.16	3.85	7.20	2.44
GAs	18	0.54	9.29	3.87	3.68	6.32	3.67	GAs	75	0.66	9.78	3.85	3.63	6.26	3.83
HARD	24	0.85	12.45	4.32	4.18	6.60	2.16	HARD	100	0.37	10.20	4.55	4.28	8.05	2.20
CHEU	24	0.64	10.26	4.05	3.83	6.16	1.82	CHEU	100	0.69	10.37	4.02	3.75	6.19	1.83
CIC	24	0.70	10.10	3.83	3.64	6.35	2.06	CIC	100	0.62	10.22	3.88	3.61	6.15	1.95
SCHE	24	0.44	12.12	4.52	4.41	7.52	2.54	SCHE	100	0.31	10.94	4.01	3.74	6.89	2.31
GAs	24	0.50	9.78	3.86	3.61	6.46	4.14	GAs	100	0.66	10.13	3.86	3.61	6.37	4.12

Table 2. Values of the ΔE_{94} error on the test set (IT8) by the models trained on the samples selected by each studied method. Fixed complete first degree polynomial. The minimum, maximum, mean, median, 90 percentile and standard deviation of ΔE_{94} error are reported. For GAs results averaged over 50 independent runs are reported.

	n. samp.	Min	Max	Mean	Median	90 prc	std dev		n. samp.	Min	Max	Mean	Median	90 prc	std dev
HARD	12	0.96	13.79	3.59	2.81	6.41	2.33	HARD	50	0.47	10.77	2.43	2.01	3.68	1.67
CHEU	12	0.67	10.35	2.53	2.11	4.21	1.59	CHEU	50	0.48	9.09	2.19	1.84	3.22	1.33
CIC	12	0.55	8.68	2.11	1.79	3.34	1.29	CIC	50	0.54	9.17	2.19	1.82	3.51	1.36
SCHE	12	1.02	9.53	2.26	1.90	3.32	1.36	SCHE	50	0.79	9.85	2.49	2.19	3.78	1.44
GAs	12	0.60	7.69	1.97	1.71	3.03	1.44	GAs	50	0.51	8.65	2.08	1.75	3.35	1.72
HARD	18	0.95	12.52	3.18	2.46	5.71	2.08	HARD	75	0.47	10.40	2.36	1.94	3.58	1.60
CHEU	18	0.67	10.97	2.54	2.06	4.10	1.68	CHEU	75	0.56	9.79	2.31	1.98	3.35	1.45
CIC	18	0.53	8.71	2.11	1.79	3.36	1.29	CIC	75	0.56	9.11	2.17	1.80	3.47	1.35
SCHE	18	0.70	7.71	2.13	1.91	2.98	1.06	SCHE	75	0.80	9.47	2.36	2.04	3.63	1.38
GAs	18	0.58	7.82	2.01	1.74	3.17	1.49	GAs	75	0.55	8.93	2.10	1.72	3.46	1.83
HARD	24	0.66	12.29	3.00	2.43	4.94	1.91	HARD	100	0.47	10.03	2.30	1.88	3.54	1.54
CHEU	24	0.54	10.12	2.38	2.00	3.47	1.49	CHEU	100	0.56	9.93	2.31	1.99	3.44	1.49
CIC	24	0.62	8.87	2.13	1.80	3.41	1.31	CIC	100	0.54	9.01	2.15	1.79	3.51	1.34
SCHE	24	0.88	10.37	2.59	2.26	3.86	1.53	SCHE	100	0.61	9.42	2.33	2.03	3.61	1.35
GAs	24	0.59	8.18	2.06	1.76	3.21	1.56	GAs	100	0.53	8.76	2.09	1.74	3.41	1.78

Table 3. Values of the ΔE_{94} error on the test set (IT8) by the models trained on the samples selected by each studied method. Fixed complete second degree polynomial. The minimum, maximum, mean, median, 90 percentile and standard deviation of ΔE_{94} error are reported. For GAs results averaged over 50 independent runs are reported.

	n. samp.	Min	Max	Mean	Median	90 prc	std dev		n. samp.	Min	Max	Mean	Median	90 prc	std dev
HARD	12	0.36	8.16	2.84	2.39	5.61	1.73	HARD	50	0.55	4.96	1.79	1.66	2.44	0.73
CHEU	12	0.71	3.91	1.67	1.57	2.40	0.56	CHEU	50	0.63	3.35	1.63	1.57	2.22	0.53
CIC	12	0.39	3.27	1.53	1.53	2.13	0.55	CIC	50	0.53	3.22	1.49	1.49	2.09	0.52
SCHE	12	0.65	2.90	1.88	1.91	2.49	0.47	SCHE	50	0.75	3.96	1.83	1.80	2.39	0.53
GAs	12	0.49	4.14	1.55	1.55	2.15	0.35	GAs	50	0.59	3.61	1.49	1.47	2.07	0.27
HARD	18	0.31	4.91	1.96	1.84	2.88	0.77	HARD	75	0.43	4.55	1.73	1.66	2.32	0.68
CHEU	18	0.81	3.52	1.66	1.61	2.23	0.50	CHEU	75	0.77	4.50	1.74	1.68	2.32	0.64
CIC	18	0.42	3.17	1.49	1.50	2.12	0.52	CIC	75	0.57	3.03	1.50	1.49	2.15	0.52
SCHE	18	0.87	2.83	1.78	1.82	2.37	0.45	SCHE	75	0.76	4.06	1.77	1.72	2.35	0.55
GAs	18	0.47	3.72	1.52	1.51	2.11	0.31	GAs	75	0.65	3.15	1.53	1.53	2.10	0.24
HARD	24	0.45	5.40	2.09	1.99	3.00	0.81	HARD	100	0.54	4.28	1.68	1.62	2.30	0.65
CHEU	24	0.69	3.33	1.61	1.55	2.16	0.48	CHEU	100	0.70	4.48	1.72	1.67	2.26	0.64
CIC	24	0.42	3.05	1.50	1.50	2.13	0.52	CIC	100	0.59	3.13	1.50	1.49	2.11	0.51
SCHE	24	0.77	4.03	2.21	2.14	3.24	0.71	SCHE	100	0.81	4.20	1.79	1.77	2.36	0.56
GAs	24	0.47	3.63	1.51	1.51	2.09	0.29	GAs	100	0.64	3.46	1.51	1.49	2.11	0.25

from the initial dataset made of 240 total samples. The polynomials that we have considered are: first degree polynomial without offset (results reported in Table 1); complete first degree polynomial (results reported in Table 2); complete second degree polynomial (results reported in Table 3); complete third degree polynomial (results reported in Table 4). Because of the non deterministic nature of GAs, we ran 50 independent executions and all the results we report were averaged over these runs. The other techniques used in this paper are deterministic, hence only one execution was needed. Following [13], the tables report the minimum, maximum, mean and median ΔE_{94} calculated on the lines of the test set (i.e. the IT8 dataset).

When the first degree polynomial without offset is considered (Table 1), GAs obtain better maximum error than all the other methods for 12, 18, 24, 75 and 100 samples; it obtains better mean error than all the other methods for 50, 75 and 100 samples and it

Table 4. Values of the ΔE_{94} error on the test set (IT8) by the models trained on the samples selected by each studied method. Fixed complete third degree polynomial. The minimum, maximum, mean, median, 90 percentile and standard deviation of ΔE_{94} error are reported. For GAs results averaged over 50 independent runs are reported.

	n. samp.	Min	Max	Mean	Median	90 prc	std dev		n. samp.	Min	Max	Mean	Median	90 prc	std dev
HARD	24	0.44	5.55	2.01	1.71	3.48	0.96	HARD	75	0.58	2.66	1.42	1.40	2.07	0.45
CHEU	24	0.57	2.88	1.45	1.48	2.08	0.49	CHEU	75	0.62	2.52	1.42	1.41	2.11	0.48
CIC	24	0.50	3.84	1.64	1.62	2.28	0.60	CIC	75	0.62	2.62	1.44	1.43	2.10	0.48
SCHE	24	0.62	6.52	2.52	2.16	4.49	1.28	SCHE	75	0.58	2.55	1.47	1.48	2.04	0.42
GAs	24	0.47	4.34	1.58	1.56	2.26	0.42	GAs	75	0.57	2.64	1.43	1.43	2.02	0.20
HARD	50	0.64	2.65	1.53	1.49	2.22	0.48	HARD	100	0.51	2.49	1.39	1.38	2.01	0.44
CHEU	50	0.58	2.84	1.42	1.39	2.23	0.52	CHEU	100	0.61	2.52	1.44	1.43	2.13	0.48
CIC	50	0.62	2.74	1.46	1.41	2.15	0.50	CIC	100	0.61	2.56	1.43	1.43	2.04	0.46
SCHE	50	0.61	3.08	1.50	1.48	2.12	0.46	SCHE	100	0.56	2.57	1.48	1.49	2.05	0.44
GAs	50	0.51	3.69	1.45	1.43	2.02	0.26	GAs	100	0.60	2.80	1.42	1.41	2.00	0.20

Table 5. Number of times (absolute and percentage values) in which the used techniques have produced the best and the worst performance on the test set (IT8), using the selected training samples

	Best	Worst	%Best	%Worst
HARD	1	10	4.55	45.45
CHEU	3	0	13.65	0
CIC	8	0	36.35	0
SCHE	0	12	0	54.55
GAs	10	0	45.45	0

Table 6. Left Part: results on the test set (IT8) returned by GAs using a fixed (randomly generated) training set and looking for the best third degree polynomial. Results averaged over 50 independent runs. The minimum, maximum, mean, median, 90 percentile and standard deviation of ΔE_{94} error are reported. Right Part: analogous results returned by GAs when optimizing both training samples and polynomial coefficients at the same time.

n. samp.	Min	Max	Mean	Median	90 prc	std dev	n. samp.	Min	Max	Mean	Median	90 prc	std dev
24	0.45	4.82	1.63	1.56	2.42	0.54	24	0.54	3.91	1.55	1.53	2.18	0.34
50	0.54	3.25	1.46	1.47	2.02	0.23	50	0.55	3.00	1.44	1.44	1.97	0.21
75	0.58	2.77	1.45	1.45	2.06	0.21	75	0.55	2.63	1.43	1.43	2.011	0.20
100	0.56	2.63	1.43	1.42	2.04	0.21	100	0.62	2.69	1.43	1.42	1.99	0.19

Table 7. Upper part: best polynomial obtained in the same 50 independent GA runs as the ones of Table 6 (left part). Lower Part: best polynomials obtained from the same 50 independent GA runs as Table 6 (right part).

n. samples	Best polynomial
24	$1+B^2+B^3+G^1B^1+G^1B^2+G^2B^1+G^3+R^1+R^1B^1+R^1B^2+R^1G^1+R^1G^1B^1+R^1G^2+R^2+R^2B^1+R^2G^1$
50	$1+B^1+B^2+B^3+G^1B^1+G^1B^2+G^2B^1+R^1+R^1B^2+R^1G^1+R^1G^1B^1+R^1G^2+R^2+R^2B^1+R^2G^1+R^3$
75	$B^2+B^3+G^1+G^1B^1+G^1B^2+G^2+G^2B^1+G^3+R^1+R^1B^1+R^1B^2+R^1G^1+R^1G^2+R^2+R^2B^1+R^2G^1+R^3$
100	$B^1+B^2+G^1B^1+G^1B^2+G^2+G^2B^1+G^3+R^1B^2+R^1G^1+R^1G^1B^1+R^1G^2+R^2+R^2B^1+R^2G^1+R^3$
24	$B^1+B^2+B^3+G^1+G^1B^1+G^2+G^3+R^1B^1+R^1B^2+R^1G^1+R^1G^2+R^2+R^2B^1+R^2G^1+R^3$
50	$1+B^1+B^2+B^3+G^1+G^1B^1+G^1B^2+G^2+G^2B^1+G^3+R^1+R^1B^2+R^1G^1+R^1G^2+R^2B^1+R^2G^1+R^3$
75	$1+B^1+B^2+B^3+G^1+G^1B^1+G^1B^2+G^2+G^2B^1+G^3+R^1+R^1B^1+R^1B^2+R^1G^1+R^1G^1B^1+R^1G^2+R^2+R^2B^1+R^2G^1+R^3$
100	$1+B^1+B^2+B^3+G^1+G^1B^1+G^1B^2+G^2+G^2B^1+G^3+R^1+R^1B^1+R^1B^2+R^1G^1+R^1G^1B^1+R^1G^2+R^2+R^2B^1+R^2G^1+R^3$

obtains better median error than all the other methods for 24, 50 and 75 samples. When the complete first degree polynomial is considered (Table 2), GAs obtain better maximum error than all the other methods for 12, 24, 50, 75 and 100 samples and it obtains both mean and median better errors than all the other methods for all the considered sampled sizes (12, 18, 24, 50, 75 and 100). When the second degree polynomial is considered (Table 3), GAs obtain better mean error than all the other methods for 18 and 50 samples and it obtains better median error than all the other methods for 50 and 100 samples. When the third degree polynomial is considered (Table 4), GAs obtain better minimum error than all the other methods for 50 and 75 samples.

To summarize the results obtained in this phase, in Table 5 we report the number of times (both in absolute and percentage values) in which the exploited techniques produced the best and the worst performances. GAs perform better than all the other considered techniques in about 45% of the experiments (and this is the maximum value reached by any of the studied techniques). Furthermore, GAs do not perform worse than all the other techniques in any of the considered experiments.

Optimizing polynomial coefficients using fixed training samples. In this phase we consider a fixed set of training samples and we want to learn the polynomial coefficients that produce the lowest mean ΔE_{94} error. In Table 6 (left part), the results produced by GAs, averaged over 50 runs, are reported. In every run a random set of samples is chosen with uniform probability from the MDC dataset, then the algorithm looks for the best third degree polynomial coefficients. Notice that the obtained results are similar to the values presented in Table 4, but the polynomials produced by the algorithm rarely contain all the terms of the complete third degree polynomial. This is a rather important result because working with a simpler polynomial is computationally less expansive than working with the full polynomial. Table 7 (upper part) reports the best polynomials obtained when 24, 50, 75 and 100 prefixed training samples have been used.

Optimizing polynomial coefficients and training samples at the same time. The goal of the third experimental phase is to learn both the training samples and the polynomial coefficients at the same time. In Table 6 (right part) results produced by GAs are shown. Finally in table 7 (lower part) the best generated polynomials are reported. Comparing the values presented in Table 6 (right part) with those displayed in Table 4, it is possible to notice a performance improvement. We have performed a t-test to understand whether the difference between the mean values shown in these two tables are significant. The t-test, performed with a confidence interval α=95% gave the following results: the difference of mean ΔE_{94} of the two distributions is statistically significant for 24 and 50 samples, while it is *not* for 75 and 100 samples.

6 Conclusions and Future Work

We have proposed a framework based on Genetic Algorithms (GAs) to optimize the most promising training samples from large datasets. The presented experimental results show that the proposed framework has better, or at least comparable, performances than a set of other computational methods defined so far for the same goal

(Hardeberg, Cheung, CIC and Schettini). Even more importantly, the proposed framework has the ability to optimize the training samples and the characterizing polynomial's coefficients at the same time (which is not feasible using the other computational methods studied here). Interestingly, the proposed framework is able to evolve polynomials that have a comparable performance with the one of full polynomials of the same degree, but using a randomly generated training set, instead of a chosen one. Furthermore, when the proposed framework evolves both the training samples and the polynomial's coefficient, it is able to find smaller average errors than in the case full polynomials are used.

Acknowledgments. L. Vanneschi and M. Castelli gratefully acknowledge project PTDC/EIACCO/103363/2008 from Fundação para a Ciência e a Tecnologia, Portugal.

References

1. Bianco, S., Gasparini, F., Schettini, R., Vanneschi, L.: An evolutionary framework for colorimetric characterization of scanners. 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. 245–254. Springer, Heidelberg (2008)
2. Bianco, S., Gasparini, F., Schettini, R., Vanneschi, L.: Polynomial modeling and optimization for colorimetric characterization of scanners. Journal of Electronic Imaging 17(4), 1–13 (2008)
3. Cheung, T.L., Westland, S.: Color selections for characterization charts. In: Proceedings of the Second European Conference on Colour in Graphics, Imaging and Vision, Aachen, Germany, pp. 116–119 (2004)
4. Cheung, T., Westland, S., Connah, D., Ripamonti, C.: Characterization of colour cameras using neural networks and polynomial transforms. Journal of Coloration Technology 120(1), 19–25 (2004)
5. Cheung, V., Westland, S., Li, C., Hardeberg, J., Connah, D.: Characterization of trichromatic color cameras by using a new multispectral imaging technique. J. Opt. Soc. Am. A 22, 1231–1240 (2005)
6. Chou, Y., Li, C., Luo, M.: A new colour selection method for characterising digital cameras. In: Proceedings of the 17th Color Imaging Conference (2009) (to appear)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
8. Hardeberg, J.Y., Brettel, H., Schmitt, F.J.M.: Spectral characterization of electronic cameras. In: Electronic Imaging: Processing, Printing, and Publishing in Color. SPIE, Zurich (1998)
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
10. Huffel, S., Vandewalle, J.: The total least squares problem: computational aspects and analysis. Society for Industrial and Applied Mathematics, Philadelphia (1991)
11. Kang, H.: Computational coolor technology, vol. PM159. SPIE Press (2006)
12. Pellegri, P., Novati, G., Schettini, R.: Training set selection for multispectral imaging systems characterization. Journal of Imaging Science and Technology 48(3), 203–210 (2004)
13. Shen, H.L., Mou, T.S., Xin, J.: Colorimetric characterization of scanners by measures of perceptual color error. Journal of Electronic Imaging 15(4), 1–5 (2006)
14. Shen, H.L., Xin, J.: Spectral characterization of a color scanner by adaptive estimation. Journal of the Optical Society of America A 21(7), 1125–1130 (2004)

New Genetic Operators in the Fly Algorithm: Application to Medical PET Image Reconstruction

Franck Patrick Vidal[1,⋆], Jean Louchet[2], Jean-Marie Rocchisani[1,3], and Évelyne Lutton[1]

[1] INRIA Saclay - Île-de-France/APIS, 4 rue J. Monod 91893 Orsay Cedex, France
[2] Artenia, 24 rue Gay-Lussac, 92320 Châtillon, France
[3] Paris XIII University, UFR SMBH & Avicenne hospital, 74 rue Marcel Cachin, 93017 Bobigny, France

Abstract. This paper presents an evolutionary approach for image reconstruction in positron emission tomography (PET). Our reconstruction method is based on a cooperative coevolution strategy (also called Parisian evolution): the "fly algorithm". Each fly is a 3D point that mimics a positron emitter. The flies' position is progressively optimised using evolutionary computing to closely match the data measured by the imaging system. The performance of each fly is assessed using a "marginal evaluation" based on the positive or negative contribution of this fly to the performance of the population. Using this property, we propose a "thresholded-selection" method to replace the classical tournament method. A mitosis operator is also proposed. It is triggered to automatically increase the population size when the number of flies with negative fitness becomes too low.

1 Introduction

Image reconstruction in tomography is an inverse problem that is ill-posed: a solution does not necessarily exist (e.g. in extreme cases of excessive noise), and the solution may not be unique. This problem can be solved as an optimisation problem, and on such cases, evolutionary algorithms have been proven efficient in general, and in particular in medical imaging [2,4,13]. We focus here on positron emission tomography (PET) reconstruction in nuclear medicine.

Nuclear medicine appeared in the 1950's [1]. Its principle is to diagnose or treat a disease by administering to patients a radioactive substance (also called tracer) that is absorbed by tissue in proportion to some physiological process. When a pathology occurs, the metabolism most of the times increases: there are more molecules in the pathology area, i.e. the radioactivity also increases.

⋆ Member of Fondation Digiteo (http://www.digiteo.fr/). Now with University of California San Diego, Rebecca and John Moores Cancer Center, Radiation Oncology, CA, USA.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 292–301, 2010.

It is possible to reconstruct slices through the human body using methods similar to those used in conventional X-ray computed tomography [7]. In nuclear medicine, this method makes use of a gamma emitter as radio-tracer. It is called Single-Photon Emission Computed Tomography (SPECT). The reconstruction allows to recover the 3D distribution of the tracer through the body.

The other main tomographic technique in nuclear medicine is PET. Here a positron emitter is used as radionuclide for labelling, rather than a single gamma emitter. Positrons are emitted with high energy (1 MeV). After interactions, a positron combines with an electron to form a positronium. Then the electron and positron pair is converted into radiations. It is the annihilation reaction, which generally produces two photons of 511 keV emitted in opposite directions. Taking advantages of this property, this radiation is detected in coincidence, i.e. using the difference in arrival times of the detected photons of each pair, and considering that each annihilation produces two photons emitted in exactly opposite directions. The line between the detectors that have been activated for a given pair of photons is called "line of response" (LOR). Prior to the reconstruction, the LOR data is often rebinned into a sinogram [5,8]. This intermediate data representation corresponds to projection data that can be used by conventional tomographic reconstruction codes. A broad overview of reconstruction methods using projection data in nuclear medicine can be found in [8,14].

The PET reconstruction methods are often divided into two classes: i) analytical methods, and ii) iterative statistical methods. Analytical methods are based on a continuous modelling and the reconstruction process consists of the inversion of measurement equations. The most frequently used is the filtered back-projection algorithm (FBP) [7]. Statistical methods are based on iterative correction algorithms. These include the most widely used techniques in SPECT and PET, such as the maximum-likelihood expectation-maximization method (ML-EM) [10] and its derivative, the ordered subset expectation-maximization algorithm (OS-EM) [6].

In a previous paper, we showed that a cooperative coevolution strategy (also called Parisian evolution) called "fly algorithm" [9] could be used in SPECT reconstruction [3]. Here, each fly corresponds to a 3D point that is emitting photons. The evolutionary algorithm is used to optimise the position of flies.

However, PET has taken over SPECT in routine clinical practice. Effort has therefore been made to propose an efficient evolutionary scheme that takes into account PET data acquisition principles [11,12], but these were still restricted to low resolution PET scanners in 2D-mode. This paper describes our current research activities aimed at providing an effective method in both 2D or fully-3D mode, and it describes recent developments, such as i) the introduction of "thresholded-selection" replacing the traditional "tournament selection" and ii) taking advantage of the thresholded-selection to increase the population size (when the number of flies, whose fitness is negative, is too low), i.e. improve the statistics of the final image. The following section gives an overview of the methodology. The results and performance of our method using numerical phantoms are presented in Section 3.

The paper ends with a conclusion that discusses the work that has been carried out and it provides directions for future work.

2 Material and Methods

2.1 Main Principles

Each individual, or fly, corresponds to a 3D point that mimics a radioactive emitter, i.e. a stochastic simulation of annihilation events is performed to compute the fly's illumination pattern. For each annihilation event, a photon is emitted in a random direction. A second photon is then emitted in opposite direction. If both photons are detected by the scanner, the fly's illumination pattern is updated. The scanner properties (e.g. detector blocks and crystals positions) are modelled, and each fly is producing an adjustable number of annihilation events. Each fly keeps a record of its simulated LORs. Therefore the result of these simulations consist of a list, per fly, of pairs of detector identification numbers that correspond to LORs. These lists are aggregated to form the population total illumination pattern.

Initially, the flies' position is randomly generated in the volume within the scanner. Using genetic operations to optimise the position of radioactive emitters, the population of flies evolves so that the population total pattern matches measured data. The final population of flies corresponds to the tracer density in the patient, i.e. the reconstructed data. Note that cross-over operations are not used in this application. In our context, the result of such an operation may lead to meaningless results, e.g. in the case of cross-over between two flies of two distinct objects (the new fly will be wrongly located in between). Only mutation and immigration (i.e. a fly is created at a random position) are used.

2.2 Fitness Metrics

The fitness metrics corresponds to a distance measurement between the simulated data and the actual data given by the imaging system. City block distance provides a good compromise between accuracy and speed. Note that smaller the population's cost is, closer the simulated data is to the actual data.

In [3], we showed that, when we were addressing the SPECT problem, if we defined the fitness of a fly as the consistency of the image pattern it generates, with the actual images, it gave an important bias to the algorithm with a tendency of the smaller objects to disappear. This is why we then introduced marginal evaluation ($F_m\,(i)$) to assess a given fly (i). It is based on the leave-one-out cross-validation method. We use a similar approach in PET:

$$\mathrm{F_m}(i) = \mathrm{dist}\,(pop, input) - \mathrm{dist}\,(pop - \{i\}\,, input) \tag{1}$$

with $\mathrm{F_m}(i)$ the marginal fitness of Fly i, $\mathrm{dist}\,(A, B)$ the city block distance between two tables A and B, pop is the set of LORs simulated by the whole population, $input$ is the set of LORs extracted from the input data, and $pop - \{i\}$

is the set of LORs simulated by the whole population without Fly i. The fitness of a given fly will only be positive when the global cost is lower (better) in presence rather than in the absence of this fly. We therefore used a fixed threshold to operate selection.

2.3 Thresholded Selection

At each iteration of the evolution loop, a fly has to be killed, and a fly may be used during the mutation. We saw in the previous section that the fly's fitness is its own contribution (positive or negative) with respect to the whole population. We take advantage of this principle as follows: i) any "bad" fly (its fitness is negative) is a candidate for death, and ii) any "good" fly (its fitness is positive) is a candidate for mutation. When a fly is killed, its LORs are removed from the total set of simulated LORs. When a new fly is created, its LORs are added. This process needs to be fast to be able to decrease the number of bad flies and increase the number of good flies as much as possible.

2.4 Mitosis

To obtain accurate, high resolution images it is necessary to use large populations of flies. However, as processing time is roughly proportional to the number of flies to be processed, we choose a simple scheme that begins with a small population, then multiply the population along the algorithm execution using a mitosis process: each fly is duplicated. Newly created flies will have their own illumination pattern.

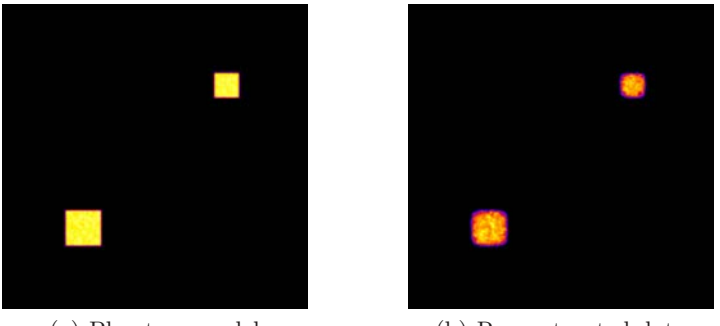
It is triggered whenever the number of flies with a negative fitness gets too low. In practice, at each step in the steady state process, one fly is chosen randomly and its fitness tested: a genetic operator will only be applied if the fly is bad. We launch the population mitosis every time 50 consecutive flies are found with a positive fitness.

3 Results

We have developed numerical phantom models to assess the reconstruction algorithm. To date, no scattering and no tissue attenuation have been considered. Whilst this is not physically correct, it allows us to test and validate our approach in the simplest cases. First, we present quantitative results in 2D-mode, both in low and high resolutions. Then, qualitative results in 3D-mode are presented using a complex object. For each test case, the initial population is 5,000 flies. When the current number of flies of the population is above a given threshold, e.g. 1^6 or 2^6, the evolution loop is stopped whenever the number of flies with a negative fitness becomes too low, i.e. the stopping criteria is similar to the mitosis criteria. 70 LORs per fly have been simulated in the 2D test cases. The probability of LORs to be detected is much higher in the 3D case. Thus only 10 LORs per fly have been simulated in the 3D case.

(a) Phantom model (b) Reconstructed data in high resolution

Fig. 1. Simulated PET System: a single ring of 72 linear blocks that include 8 crystals; two boxes ($7 \times 7 \times 0.4$ cm^3 and $10 \times 10 \times 0.4$ cm^3) with the same radioactivity concentrations ($\sim$ 930.000 counts/ml)



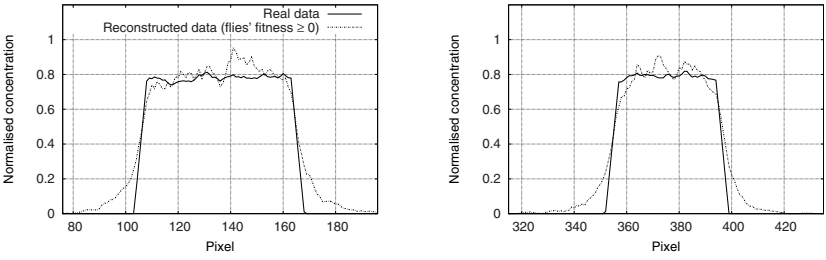
(a) Phantom model (b) Reconstructed data

Fig. 2. Slices (512×512 pixels) through the cubes

3.1 2D-Mode

Test 1: large objects with different sizes and similar radioactivity concentration. The purpose of this test was to assess the ability of the algorithm to retrieve relatively large objects, whose sizes are different, but with the same radioactivity concentration. Fig. 1 shows the simulated set up. The phantom is made of two boxes ($7 \times 7 \times 0.4$ cm^3 and $10 \times 10 \times 0.4$ cm^3) with the same radioactivity concentrations ($\sim$ 930.000 counts/ml). The simulated PET system is made of a single ring of 72 linear blocks that include 8 crystals. To evaluate the results, a 512×512 pixel slice is produced (see Fig. 2). Note that the typical image size in PET is 128×128 pixels. The slices are post-filtered using a gaussian convolution kernel, then linearly rescaled between zero and one. Profiles in this reconstructed image are compared to corresponding profiles in the phantom data (see Fig. 3). For both boxes, the reconstructed data seems to be close to the input data. Full width at half maximum (FWHM) is also measured to quantify errors (see Table 1). These results show that our evolutionary scheme is able to accurately recover the width of our test objects.

Test 2: small objects with different sizes and radioactivity concentrations. This test case has been designed to assess the ability of our algorithm

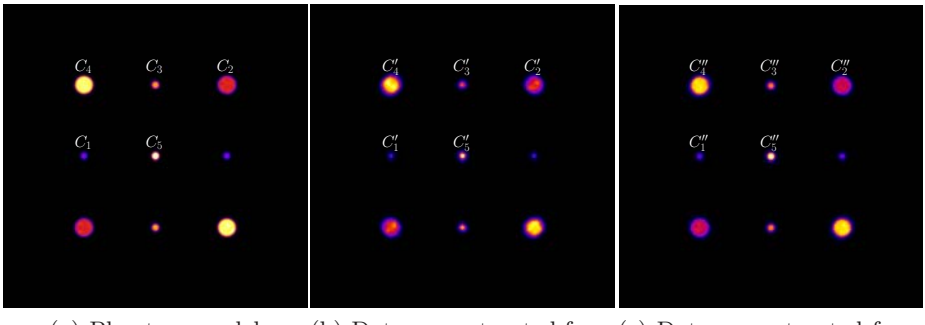


(a) Cube size: $10 \times 10 \times 0.4$ cm^3 (b) Cube size: $7 \times 7 \times 0.4$ cm^3

Fig. 3. Profiles extracted from Fig. 2

Table 1. FWHM estimated from Fig. 3

FWHM from phantom model (in mm)	FWHM from slice (in mm)	Relative difference (in %)
71	72	1
99	99	0



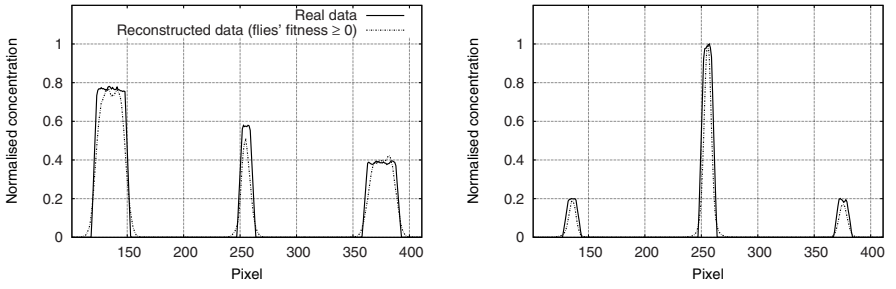
(a) Phantom model (b) Data reconstructed from low resolution scanner (c) Data reconstructed from high resolution scanner

Fig. 4. Slices (512×512 pixels) through the cylinders

to detect small objects, and their relative radioactivity concentrations. Fig. 4(a) shows nine cylinders having two different radii (1 cm and 2.5 cm) and five different radioactivity concentrations ($C_1 = 114,590$ count/ml, $C_2 = 2C_1$, $C_3 = 3C_1$, etc.). A low resolution PET system has first been considered. It is made of a single ring of 72 linear blocks that include only 1 crystal.

To evaluate the results, a 512×512 pixel slice is produced once again (see Fig. 4(b)). The reconstructed data appears to be visually close to the input data. In particular, the size and concentration of cylinders are visually well preserved.

To estimate the diameter of each cylinder, horizontal profiles have been extracted so that they crossed the cylinders in their respective centre (see Fig. 5).



(a) 1st line (b) 2nd line

Fig. 5. Profiles extracted from Fig. 4(b)

Table 2. FWHM estimated from Fig. 5 (using a low resolution PET system), and Fig. 6 (using a high resolution PET system)

Object	FWHM from phantom model (in mm)	FWHM in Fig. 5 (in mm)	Relative difference in Fig. 5 (in %)	FWHM in Fig. 6 (in mm)	Relative difference in Fig. 6 (in %)
1	19	13	31.6	18	5.7
2	49	43	12.2	48	2.3
3	19	14	26.3	18	6.8
4	49	44	10.2	47	3.2
5	19	12	36.8	17	8.6

As the lower profiles are symmetrically similar to the upper profiles, they are not plotted here. FWHM is measured once again to quantify errors (see Table 2). Let Object i be the cylinder whose concentration is C_i in the phantom model and C'_i in the reconstructed slice. Whilst the profiles in the reconstructed slice seem to match respective profiles in the phantom model, error measurements in FWHM are relatively high for the smallest cylinders (up to 35%). To investigate the influence of the reconstructed slice resolution with respect to the low spatial resolution of the PET system, the test case presented in the next section makes used of similar objects and a PET scanner with higher spatial resolution.

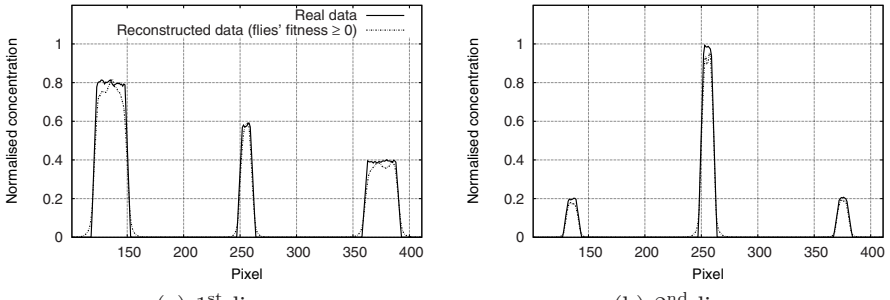
To assess the validity of the radioactivity concentration within cylinders, the average value at the centre of each cylinder has been measured in Fig. 4(b) (see Table 3). We compare the respective ratio of the different concentrations to the lower value (C'_1), so that we can compare the reconstructed values with the theoretical values. In theory, we should get $C'_2 = 2C'_1$, $C'_3 = 3C'_1$, etc. Table 3 shows that the relative concentrations have been preserved in the reconstructed slice. However, the maximum relative error is about 16.50%.

Test 3: higher scanner resolution. The previous test case shows that our algorithm is able to retrieve the respective size of objects and their respective concentration. However, relative errors can be as high as 35% for the FWHM

Table 3. Relative radioactivity concentration estimated from Fig. 4(b) (using a low resolution PET system), and Fig. 4(c) (using a high resolution PET system)

Object	Relative concentration in Fig. 4(b)	Relative error in Fig. 4(b) (in %)	Relative concentration in Fig. 4(c)	Relative error in Fig. 4(c) (in %)
1	C'_1	N/A	C''_1	N/A
2	$2.13 \times C'_1$	6.5	$2.17 \times C''_1$	8.4
3	$2.67 \times C'_1$	16.5	$3.19 \times C''_1$	9.5
4	$3.80 \times C'_1$	10.0	$4.40 \times C''_1$	19.9
5	$5.02 \times C'_1$	1.0	$5.35 \times C''_1$	17.5

and 16,5% for the concentration. In this test case, similar objects have been simulated. The size of crystals has been reduced so that their width matches the width of real crystals. A similar methodology is used to assess the results: i) Fig. 4(c) shows a 512×512 pixel slice that has been reconstructed, ii) profiles have been extracted (see Fig. 6), iii) FWHM estimated (see Table 2), and iv) the concentrations assessed (see Table 3). These results show that using a high



(a) 1st line (b) 2nd line

Fig. 6. Profiles extracted from Fig. 4(c)

resolution PET scanner reduced the maximum error in object size by a factor of 4. This is due to the improvement of the spatial resolution. On the other hand, errors in radioactivity concentration have not been reduced.

3.2 3D-Mode

The last case has been performed in fully-3D, i.e. the PET imaging system is made of a stack of detector rings. For a coincidence event, the two photons of a LOR can be detected onto different rings. Only visual results are presented here.

A complex 3D shape is used in this test. The simulation is performed using a polygon mesh (here we use the dragon model from *The Stanford 3D Scanning Repository*, http://graphics.stanford.edu/data/3Dscanrep/, last access 17 Jan

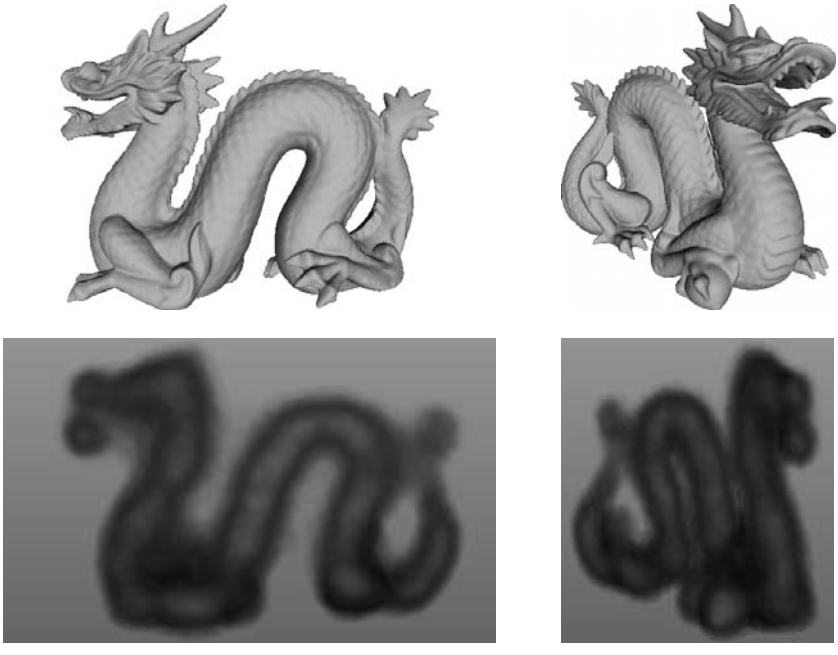


Fig. 7. Top row: simulated object; bottom row: volume rendering of the reconstruction

2010) that is uniformly filled with radio-tracers (see top row in Fig. 7). Then, LORs are recorded in fully-3D mode. Finally, we run our evolutionary reconstruction scheme. Note that the reconstruction algorithm is similar in both 2D and 3D modes. The only difference is the geometrical property of the simulated PET scanner. The bottom row in Fig. 7 presents the reconstructed dataset after volume rendering. One can visually distinguish the shape of the dragon from the population of flies.

4 Conclusion

It may occur that complex applications fuel fundamental technical developments. In the research presented here, we addressed a complex problem that had never been approached in the past using evolutionary computing, by transposing the Fly Algorithm technique originally developed in a stereovision context. We then faced several difficult issues which encouraged the development of new tools that can probably be used into other application fields in evolutionary computing. Using the 'marginal fitness' concept opened the way to using a simplified thresholded selection, which in turn allowed to introduce the mitosis operator that duplicates the population whenever the proportion of individuals with a negative contribution to the global fitness becomes too low, thus periodically reviving the efficiency of the classical operators (mutation and immigration).

Preliminary results on tests objects show the validity of this approach in both 2D and fully-3D modes. In particular, the size of objects, and their relative concentrations can be retrieved in the 2D mode. In fully-3D, complex shapes can be reconstructed.

To date, only true coincidence events have been considered. Further work will therefore include the use of more realistic input data (including random events and scattering), which will finally lead to implement the correction of scattering within our algorithm. A comparison study against ML-EM and/or OS-EM methods will also need to be conducted.

References

1. Badawi, R.D.: Nuclear medicine. Phys. Educ. 36(6), 452–459 (2001)
2. Bosman, P.A.N., Alderliesten, T.: Evolutionary algorithms for medical simulations: a case study in minimally-invasive vascular interventions. In: Workshops on Genetic and Evolutionary Computation 2005, pp. 125–132 (2005)
3. Bousquet, A., Louchet, J., Rocchisani, J.M.: Fully three-dimensional tomographic evolutionary reconstruction in nuclear medicine. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (eds.) EA 2007. LNCS, vol. 4926, pp. 231–242. Springer, Heidelberg (2008)
4. Cagnoni, S., Dobrzeniecki, A.B., Poli, R., Yanch, J.C.: Genetic algorithm-based interactive segmentation of 3D medical images. Image Vision Comput. 17(12), 881–895 (1999)
5. Fahey, F.H.: Data acquisition in PET imaging. J. Nucl. Med. Technol. 30(2), 39–49 (2002)
6. Hudson, H.M., Larkin, R.S.: Accelerated image reconstruction using ordered subsets of projection data. IEEE Trans. Med. Imaging 13(4), 601–609 (1994)
7. Kak, A.C., Slaney, M.: Principles of computerized tomographic imaging. Society of Industrial and Applied Mathematics (2001)
8. Lewitt, R.M., Matej, S.: Overview of methods for image reconstruction from projections in emission computed tomography. Proc. of IEEE 91, 1588–1611 (2003)
9. Louchet, J.: Stereo analysis using individual evolution strategy. In: International Conference on Pattern Recognition 2000, p. 1908 (2000)
10. Shepp, L.A., Vardi, Y.: Maximum likelihood reconstruction for emission tomography. IEEE Trans. Med. Imaging 1(2), 113–122 (1982)
11. Vidal, F.P., Lazaro-Ponthus, D., Legoupil, S., Louchet, J., Lutton, E., Rocchisani, J.: Artificial evolution for 3D PET reconstruction. In: Artificial Evolution 2009. LNCS. Springer, Heidelberg (2009) (to appear)
12. Vidal, F.P., Louchet, J., Lutton, E., Rocchisani, J.: PET reconstruction using a cooperative coevolution strategy in LOR space. In: IEEE Medical Imaging Conference 2009 (2009) (to appear)
13. Völk, K., Miller, J.F., Smith, S.L.: Multiple network CGP for the classification of mammograms. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 405–413. Springer, Heidelberg (2009)
14. Zaidi, H. (ed.): Quantitative Analysis in Nuclear Medicine Imaging. Springer, US (2006)

Chaotic Hybrid Algorithm and Its Application in Circle Detection

Chun-Ho Wu[1], Na Dong[1,2], Wai-Hung Ip[1], Ching-Yuen Chan[1], Kei-Leung Yung[1], and Zeng-Qiang Chen[2]

[1] Department of ISE, The Hong Kong Polytechnic University, HungHom, Kln, Hong Kong
[2] Department of Automation, Nankai University, Tianjin, 300071, China
jack.wu@polyu.edu.hk, dongna1110@hotmail.com,
{mfwhip,mfcychan,mfklyung}@inet.polyu.edu.hk,
chenzq@nankai.edu.cn

Abstract. An evolutionary circle detection method based on a novel Chaotic Hybrid Algorithm (CHA) is proposed. The method combines the strengths of particle swarm optimization, genetic algorithms and chaotic dynamics, and involves the standard velocity and position updating rules of PSO with the ideas of GA selection, crossover and mutation. In addition, the notion of species is introduced into the proposed CHA to enhance its performance in solving multimodal problems. The effectiveness of the Species based Chaotic Hybrid Algorithm (SCHA) is proven through simulations and benchmarking, and finally, it is successfully applied to solve circle detection problems.

Keywords: Circle Detection, Chaos, PSO, GA, Multimodal Optimization.

1 Introduction

Genetic Algorithms (GA) and Particles Swarm Optimization (PSO) are both population based algorithms that have proven to be successful in solving a variety of difficult problems. However, both models have strengths and weaknesses. Comparisons between GAs and PSOs have been performed by Eberhart and Angeline and both conclude that a hybrid of the standard GA and PSO models could lead to further advances [1, 2]. Recently, a hybrid GA/PSO algorithm, Breeding Swarms (BS), combining the strengths of GA with those of PSO, was proposed by Matthew and Terence [3]. The performance of BS is competitive with both the GA and PSO, and was able to locate an optimum significantly faster than either GA or PSO. In a GA, if an individual is not elite or selected for crossover, the individual's information is lost. However, without a selection operator, PSOs usually waste resources on poor individuals. The hybrid algorithm combines the standard velocity and position updating rules of PSO with the ideas of GA selection, crossover and mutation. The operations inherited from GA facilitate a search globally, but not exactly, while the interactions of PSO effectuate a search for an optimum. In order to improve the whole performance and to enhance the GA's operations in terms of searching ability, the notion of chaos is introduced into the initialization and replaces the ordinary GA mutation. Chaos is a kind

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 302–311, 2010.

of characteristic of nonlinear systems and chaotic motion can traverse every state in a certain region by its own regularity, and nowadays has been applied in different fields [4, 5]. Due to the unique ergodicity and special ability in avoiding being trapped in local optima, the performance of chaos search is much higher than some other stochastic algorithms [6].

Multimodal optimization is used to locate all the optima within the searching space, rather than one and only one optimum, and has been extensively studied by many researchers [7]. Many algorithms based on a large variety of different techniques have been proposed in the literature. Among them, 'niches and species' and a fitness sharing method [8] were introduced to overcome the weakness of traditional evolutionary algorithms for multimodal optimization. Here, the notion of species [9] is combined with the proposed Chaotic Hybrid Algorithm (CHA) to solve multimodal problems, and it is then put into use in the application of circle detection.

The circle detection problem has attracted many researchers and most of them apply Hough transform based techniques. For examples, Lam and Yuen [10] proposed an approach which is based on hypothesis filtering and Hough transforms to detect circles. Rosin and Nyongesa [11] suggested a soft computing approach to shape classification. In this paper, the three-edge-point circle representation is adopted [12], which can reduce the search space by eliminating infeasible circle locations in the captured images.

2 GA/PSO Hybrid Algorithms

Since the paper presents a new version of a combined evolutionary search method, a brief literature review on the GA/PSO hybrid algorithm is included. The proposed CHA will then be introduced.

2.1 Breeding Swarms (BS)

The hybrid algorithm combines the standard velocity and position updating rules of PSO with the ideas of selection, crossover and mutation [3]. An additional parameter, the breeding ratio (φ), determines the proportion of the population which undergoes breeding (selection, crossover and mutation) in the current generation. Values for the breeding ratio parameter range within (0.0 : 1.0). In each generation, after the fitness values of all the individuals in the same population are calculated, the bottom portion $(N \cdot \varphi)$, where N is the population size, is discarded and removed from the population. The remaining individuals' velocity vectors are updated and acquire new information from the population. The next generation is then created by updating the position vectors of these individuals to refill $(N \cdot (1 - \varphi))$ individuals in the next generation. The $(N \cdot \varphi)$ individuals, which are required to refill in the population, are selected from the preserved individuals. The velocity of each individual is updated by undergoing the Velocity Propelled Averaged Crossover (VPAC) and mutation, and the above mentioned process is repeated in each iteration. The crossover operator, VPAC, incorporates the PSO velocity vector. The goal is to create two child particles whose positions are between the parents' positions, but accelerated away from the

parent's current directions (negative velocity) in order to increase diversity in the population. Equation (1) shows how the new child position vectors are calculated:

$$\begin{aligned} c_1(x_i) &= (p_1(x_i) + p_2(x_i))/2.0 - \varepsilon_1 p_1(v_i) \\ c_2(x_i) &= (p_1(x_i) + p_2(x_i))/2.0 - \varepsilon_2 p_2(v_i) \end{aligned} \tag{1}$$

where, $c_1(x_i)$ and $c_2(x_i)$ are the positions of children 1 and 2 in dimension *i*, respectively. $p_1(x_i)$ and $p_2(x_i)$ are the positions of parents 1 and 2 in dimension *i*, respectively. $p_1(v_i)$ and $p_2(v_i)$ are the velocities of parents 1 and 2 in dimension *i*, respectively. ε is a uniform random variable in the range [0.0 : 1.0]. The child particles retain their parent's velocity vector, $c_1(\vec{v}) = p_1(\vec{v})$ and $c_2(\vec{v}) = p_2(\vec{v})$ .The previous best vector is set to the new position vector of the child.

2.2 Chaotic Hybrid Algorithm (CHA)

In this paper, a new hybrid algorithm based on GA and PSO, CHA, is proposed. In order to improve the whole performance and to let the operations inherited from GA have a better performance in the global search situation, the notion of chaos is introduced into the initialization and as a replacement of the mutation process. In this paper, the tent map is used to generate chaos variables. It shows the outstanding advantages and higher iterative speed [13] which is more suitable for the uniform distribution function in the interval [0, 1]. The tent map is defined by:

$$z_{n+1} = \mu\left(1 - 2\left|z_n - 0.5\right|\right), 0 \le z_0 \le 1 \text{ , } n = 0,1,2,\cdots \tag{2}$$

where $u \in (0,1)$ is the bifurcation parameter. Specifically, when $\mu=1$, the tent map exhibits entirely chaotic dynamics and ergodicity in the interval [0, 1]. The tent map chaotic dynamics have been used to initialize the particle swarm. Firstly, the tent map($\mu=1$)is used to generate the chaos variables, and rewriting Equation (2), gives:

$$z_j^{(i+1)} = \mu\left(1 - 2\left|z_j^{(i)} - 0.5\right|\right), \ j = 1,2,\cdots,D \tag{3}$$

where z_j denotes the j_{th} chaos variable, and *i* denotes the chaos iteration number. Set *i=0* and generate *D* chaos variables by Equation (3). After that, let *i=1, 2,..., m* in turn, and generate the initial swarm. Then, the above chaos variable $z_j^{(i)}, i = 1,2,\cdots,m$, will be mapped into the search range of the decision variable:

$$x_{ij} = x_{\min,j} + z_j^{(i)}\left(x_{\max,j} - x_{\min,j}\right), \ j = 1,2,\cdots,D \tag{4}$$

defining:

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{iD}), \ i = 1,2,\ldots,m. \tag{5}$$

and then, the chaos initialized particle swarm can be obtained.

The above chaos initialization method is used to initialize the whole swarm, and unlike the BS, in the proposed CHA, the GA mutation process is not operated by the traditional mutation method of changing the '0, 1' sequence, but is replaced by the chaos re-initialization approach. That is, when an individual is chosen to do the mutation, it is re-initialized by the chaos initialization. For clarity, the flow of the proposed chaotic method is illustrated in Fig. 1, where $n = (N \cdot (1 - \varphi))$.

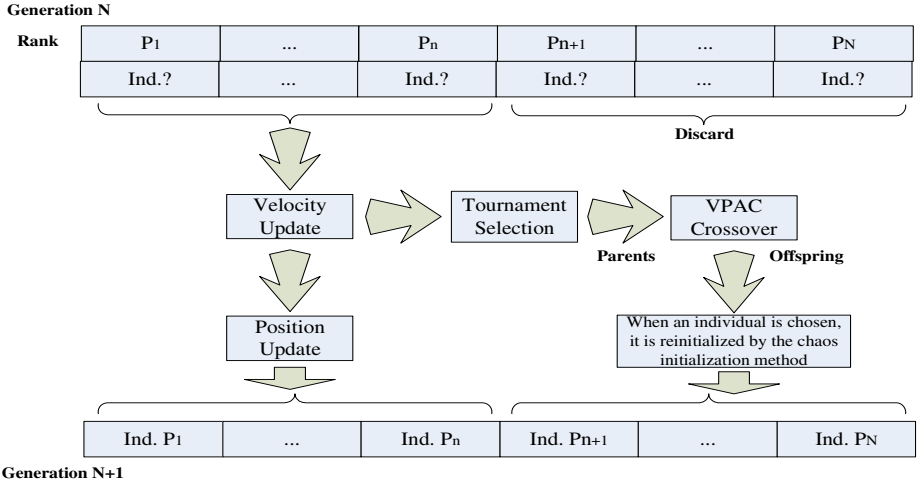


Fig. 1. Flow Diagram of the Chaotic Hybrid Algorithm

3 Species Based Chaotic Hybrid Algorithm (SCHA)

Multimodal optimization is used to locate all the optima within the search space, rather than one and only one optimum, and has been extensively studied by many researchers. Many algorithms based on a large variety of different techniques have been proposed in the literature. Recently, a speciation based particle swarm optimization (SPSO) method [9] was introduced to solve multimodal problems. The SPSO aims to identify multiple species within a population, and determines the neighborhood best for each species. The multiple species are produced adaptively, in parallel, and are used to optimize multiple optima. In this paper, the notion of species is incorporated into the proposed CHA, and to generate a new Species based Chaotic Hybrid Algorithm (SCHA).

3.1 Procedure of SCHA

The notion of species has been incorporated with CHA. A species can be defined as a group of individuals sharing common attributes according to some similarity metric. This similarity metric could be based on the Euclidean distance for genotypes using a real coded representation. The smaller the Euclidean distance between two individuals, the more similar they are. The definition of species also depends on another

parameter γ_s, which denotes the radius measured in the Euclidean distance from the center of a species to its boundary. The center of a species, the species seed, is always the fittest individual in the species. All particles that fall within the γ_s distance from the species seed are classified as the same species. The particles start searching for the optimum of a given objective function by moving through the search space at a random initial position. The manipulation of the swarm can be represented by Equations (6) and (7). Equation (6) updates the particle velocity and Equation (7) updates each particle's position in the search space, where p_{id} is the personal best position and $lbest_i$ is the neighborhood best of particle i.

$$v_{id}^{k+1} = \omega v_{id}^{k} + c_1 r_1 (p_{id}^{k} - x_{id}^{k}) + c_2 r_2 (lbest_{id}^{k} - x_{id}^{k}) \tag{6}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{7}$$

Once the species seeds have been identified from the population, one can then allocate each seed to be the *lbest* to all the particles in the same species at each iteration step. The whole population can then be divided into several sub-species when the species seeds have been identified from the population. At each iteration step, all particles within each sub-species are updated by the proposed CHA method.

3.2 Testing Results

In order to test the proposed SCHA's ability to locate multiple maxima, Himmelblau's function is introduced as a test function:

$$F(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2 \tag{8}$$

It has two variables x and y, where $-6 \le x, y \le 6$. This function has four identical global maxima which all equal to 200. For comparison, the notion of species is also incorporated into the BS algorithm [3]. Its procedure is similar to the proposed SCHA, and it just modifies the particles' updating rules by the BS method. All the trial tests are coded in MATLAB and executed on an Intel (R) 3.0 GHz CPU with a 2G RAM desktop computer. In the simulation, the parameter settings are: swarm size 120, the inertia weight ω is linearly decreased from 0.9 to 0.2, the cognitive coefficients $c1 = c2 = 2$, tournament size of 3, mutation rate is reduced linearly each generation from 1.0 to 0.1, and γ_s is given by a moderate value of 3.0, for both the Species based BS algorithm and the proposed SCHA. The maximum number of iterations is set to 1500 and the simulation is conducted by running the Species based BS and the SCHA 30 times. The maximal, the minimal and the mean iteration required to locate the maxima are shown in Table 1, and the maximal, the minimal and the mean of the total time to locate the maxima are shown in Table 2. From Tables 1 and 2, it can be seen clearly that the proposed SCHA is much more efficient and more effective than the Species based BS. The optima searching process has been improved over 77% using the proposed SCHA, in terms of averaged run time. Thus, the SCHA is a better means for solving multi-hump problems.

Table 1. The maximal, the minimal and the mean iteration steps to locate all the maxima

Algorithm	Min. Iteration	Max. Iteration	Successful Rate* (%)	Iteration (Mean & Std. err.)
Species based BS	199	1500	43.33	1041.4±582.67
SCHA	163	278	100	201.9±25.12

* Successful rate means how many times that all maxima can be successfully located out of 30 runs.

Table 2. The maximal, the minimal and the mean time to locate the maxima out of 30 runs

Algorithm	Min. Run Time (s)	Max. Run Time (s)	Successful Rate* (%)	Time (Mean & Std. err.)
Species based BS	0.69	3.98	43.33	2.81±1.45
SCHA	0.52	1.22	100	0.62±0.12

* Successful rate means how many times that all maxima can be successfully located during 30 runs.

4 Application of the SCHA in Circle Detection

In this section, we applied the proposed SCHA method to the application of circle detection. In this paper, the three-edge-point circle representation method [12] is adopted, which can reduce the search space by eliminating infeasible circle locations.

4.1 Circle Representation and Fitness Evaluation

Each particle composes three edge points which represent a circle (C). In this representation, edge points are stored as an index to their relative position in the edge array V of the image. This will encode an individual as the circle that passes through the three points v_i v_j and v_k. Each C is represented by the three parameters x_0, y_0 and r, with (x_0, y_0) being the (x, y) coordinates of the center of the circle and r being its radius. One can compute the equation of the circle passing through 3 edge points as:

$$(x-x_0)^2+(y-y_0)^2=r^2 \tag{9}$$

with:

$$x_0=\frac{\begin{vmatrix} x_j^{\,2}+y_j^{\,2}-(x_i^{\,2}+y_i^{\,2}) & 2(y_j-y_i) \\ x_k^{\,2}+y_k^{\,2}-(x_i^{\,2}+y_i^{\,2}) & 2(y_k-y_i) \end{vmatrix}}{4((x_j-x_i)(y_k-y_i)-(x_k-x_i)(y_j-y_i))} \tag{10}$$

$$y_0=\frac{\begin{vmatrix} 2(x_j-x_i) & x_j^{\,2}+y_j^{\,2}-(x_i^{\,2}+y_i^{\,2}) \\ 2(x_k-x_i) & x_k^{\,2}+y_k^{\,2}-(x_i^{\,2}+y_i^{\,2}) \end{vmatrix}}{4((x_j-x_i)(y_k-y_i)-(x_k-x_i)(y_j-y_i))} \tag{11}$$

The shape parameters (for the circle, $[x_0, y_0, r]$) can then be represented as a transformation T of the edge vector indexes i, j, k. T is the transformation composed of the previous computations for x_0, y_0 and r.

$$[x_{0,} y_0, r] = T(i, j, k) \tag{12}$$

In order to compute the fitness value of a single C, the test set for the points is $S = \{s_1, s_2, ..., s_{Ns}\}$ with N_s test points where the existence of an edge border will be sought. The test point set S is generated by the uniform sampling of the shape boundary. The N_s test points are generated around the circumference of the candidate circle. Each point s_i is a 2D-point where its coordinates (x_i, y_i) are computed as follows:

$$x_i = x_0 + r \cdot \cos\frac{2\pi i}{Ns} \quad , \quad y_i = y_0 + r \cdot \sin\frac{2\pi i}{Ns} \tag{13}$$

The fitness function $F(C)$ accumulates the number of expected edge points (i.e. the points in the set S) that are actually present in the edge image. That is:

$$F(C) = (\sum_{i=0}^{N_s - 1} E(x_i, y_i)) / N_s \tag{14}$$

with $E(x_i, y_i)$ being the evaluation of the edge features in the image coordinates (x_i, y_i) and N_s being the number of pixels in the perimeter of the circle corresponding to the individual C under test. As the perimeter is a function of the radius, this serves as a normalization function with respect to the radius. That is, $F(C)$ measures the completeness of the candidate circle and the objective is then to maximize $F(C)$ because a larger value implies a better circularity.

4.2 Simulation Results

To test the proposed SCHA method, 3 synthetic images of 640 x 480 pixels have generated, with randomly located circles (Fig. 2). Results of interest are the center of the circle position and its diameter. The algorithms have been run 30 times for each test. The species radius γ_s is defined as the distance between each two circles' centers. For the three test images, the exact number of circles and the distances between their centers are known, and the species radius γ_s is set normally to a value smaller than the distance between the two closest circles. The parameters in the setup, for both the Species based BS and the SCHA, are: swarm size 120, inertia weight ω is linearly decreased from 0.9 to 0.2, cognitive coefficients $c1 = c2 = 2$, tournament size of 3, mutation ratio is reduced linearly each generation from 1.0 to 0.1, and maximum generation of 5000. The simulation results are shown in Tables 3, 4 and 5.

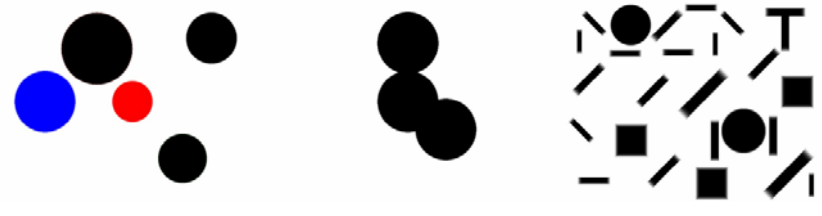


Fig. 2. Three synthetic testing images: Five circles (Left); Three merged circles (Middle); Two circles with mixed shapes (Right)

Table 3. The simulation results of the five-circle testing image

Methods	Species based BS	SCHA
Successful rate	100%	100%
Time (mean & std. err)	0.63s±0.29	0.52s±0.21
Time (min.)	0.250	0.172

* Successful rate means how many times that all 5 circles can be successfully located during 30 runs

Table 4. The simulation results on the three-merged-circle testing image

Methods	Species based BS	SCHA
Successful rate	100%	100%
Time (mean & std. err)	0.23s±0.07	0.14s±0.05
Time (min.)	0.141s	0.109s

* Successful rate means how many times that all 3 circles can be successfully located during 30 runs

Table 5. The simulation results on the two-circle with mixed shapes testing image

Methods	Species based BS	SCHA
Successful rate	76.7%	100%
Time (mean & std. err)	26.25s±21.10	16.66s±13.46
Time (min.)	7.547s	1.516s

* Successful rate means how many times that both 2 circles can be successfully located during 30 runs

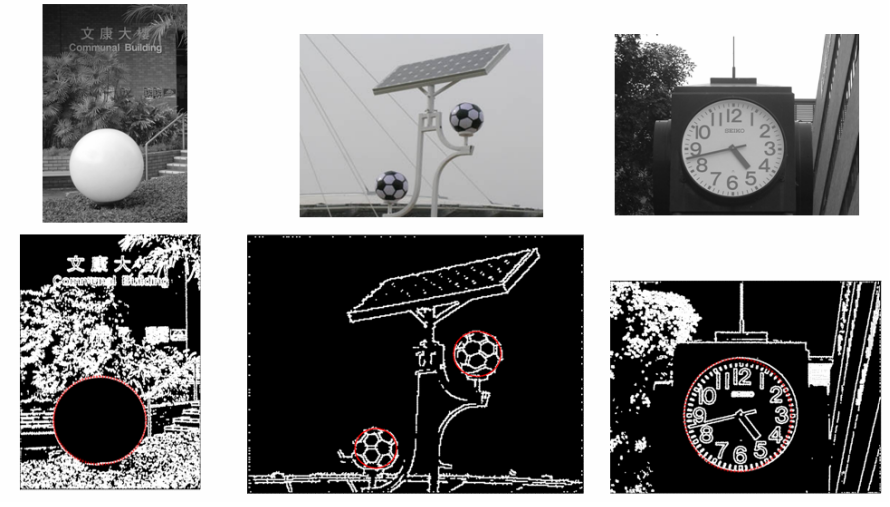


Fig. 3. Three captured natural images and the circle detection results

From Table 3 and Table 4, it can be seen that both the Species based BS and the SCHA can achieve 100% successful rate to locate all the circles, but the proposed SCHA method is more efficient than the Species based BS, and gives more exact positions and diameters of the circles, in most cases. From Table 5, the advantages of the proposed SCHA are more obvious. It can take less time and has a 100% successful rate,

while the Species based BS can only achieve 76.7%. By using the SCHA, the average error for localization is 0.08 pixels and maximum error (the worst case) is 0.94. Thus, the algorithm identifies circles in a sub-pixel level.

In order to fully illustrate the effectiveness of the proposed SCHA method, three natural images are introduced in Fig. 3 and the corresponding circle detection results are shown below. Since the ground truth reference data is not available, the results have been statistically analyzed for comparison. The mean and standard deviation for the detected parameters for a 30 run test for each image have been computed. The standard deviation is within the 0.96 pixels range and all circles are detected with a 100% successful rate. These are positive results because the proposed algorithm is robust in translation, scale in the field of view, existing noise and irregularities in shape.

5 Conclusions

In this paper, a novel Chaotic Hybrid Algorithm (CHA) is proposed, combining the strengths of chaos dynamics, particle swarm optimization and genetic algorithms. The notion of species is also introduced into the novel CHA to solve multimodal problems. The proposed Species based Chaotic Hybrid Algorithm (SCHA) is compared to the Species based BS algorithm, in evolving solutions to a standard Himmelblau function. Results show that the proposed algorithm is highly competitive, often outperforming the other method. The effectiveness of the SCHA method is proven through simulations, and finally, after proving its effectiveness through simulations on three synthetic images, it was successfully applied to solve single and multiple circles detection problems in natural images. According to the performance of the proposed method, future work will be directed towards applying it to more challenging cases, for examples, PCB components and ball grid array inspection. The SCHA will also be extended to other imaging problems.

Acknowledgments. Our gratitude is extended to the research committee and the Dept. of ISE of the Hong Kong Polytechnic University for support in this project (GYG44). This work was also supported in part by the Application Base and Frontier Technology Research Project of Tianjin, China, under 08JCZDJC21900.

References

1. Angeline, P.: Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)
2. Eberhart, R., Shi, Y.: Comparison between Genetic Algorithms and Particle Swarm Optimization. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 611–616. Springer, Heidelberg (1998)
3. Settles, M., Soule, T.: Breeding Swarms: a GA/PSO Hybrid. In: The Genetic and Evolutionary Computation Conference (GECCO 2005), pp. 161–168. ACM, New York (2005)
4. Wong, K.W., Kwok, S.H., Law, W.S.: A Fast Image Encryption Scheme Based on Chaotic Standard Map. Phys. Lett. A. 372, 2645–2652 (2008)

5. Lu, Z., Shieh, L.S., Chen, G.R.: On Robust Control of Uncertain Chaotic Systems: a Sliding Mode Synthesis via Chaotic Optimization. Chaos, Solitons Fractals 18, 819–827 (2003)
6. Li, B., Jiang, W.S.: Optimizing Complex Functions by Chaos Search. Int. J. Cybern. Syst. 29(4), 409–419 (1998)
7. Petalas, Y.G., Antonopoulos, C.G., Bountis, T.C., Vrahatis, M.N.: Detecting Resonances in Conservative Maps Using Evolutionary Algorithms. Phys. Lett. A 373, 334–341 (2009)
8. Goldberg, D.E., Richardson, J.: Genetic Algorithms with Sharing for Multimodal Function Optimization. In: Proc. 2nd International Conference on Genetic Algorithms (ICGA), pp. 41–49 (1987)
9. Parrott, D., Li, X.: Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation. IEEE Trans. Evol. Comput. 10(4), 440–457 (2006)
10. Lam, W., Yuen, S.: Efficient Techniques for Circle Detection Using Hypothesis Filtering and Hough Transform. IEE Proc. Visual Image Signal Proc. 143(5), 292–300 (1996)
11. Rosin, P.L., Nyongesa, H.O.: Combining Evolutionary, Connectionist, and Fuzzy Classification Algorithms for Shape Analysis. In: Cagnoni, S., et al. (eds.) EvoWorkshops 2000. LNCS, vol. 1803, pp. 87–96. Springer, Heidelberg (2000)
12. Victor, A.R., Carlos, H.G.C., Arturo, P.G., Raul, E.S.Y.: Circle Detection on Images Using Genetic Algorithms. Pattern Recognit. Lett. 27(6), 652–657 (2006)
13. Zhang, H., Shen, J.H., Zhang, T.N., Li, Y.: An Improved Chaotic Particle Swarm Optimization and Its Application in Investment. In: Proc. International Symposium on Computational Intelligence and Design, vol. 1, pp. 124–128 (2008)

Content-Based Image Retrieval of Skin Lesions by Evolutionary Feature Synthesis

Lucia Ballerini[1], Xiang Li[1], Robert B. Fisher[1], Ben Aldridge[2], and Jonathan Rees[2]

[1] School of Informatics, University of Edinburgh, UK
x.li-29@sms.ed.ac.uk, lucia.ballerini@ed.ac.uk, rbf@inf.ed.ac.uk
[2] Dermatology, University of Edinburgh, UK
ben.aldridge@ed.ac.uk, jonathan.rees@ed.ac.uk

Abstract. This paper gives an example of evolved features that improve image retrieval performance. A content-based image retrieval system for skin lesion images is presented. The aim is to support decision making by retrieving and displaying relevant past cases visually similar to the one under examination. Skin lesions of five common classes, including two non-melanoma cancer types, are used. Colour and texture features are extracted from lesions. Evolutionary algorithms are used to create composite features that optimise a similarity matching function. Experiments on our database of 533 images are performed and results are compared to those obtained using simple features. The use of the evolved composite features improves the precision by about 7%.

1 Introduction

Research in content-based image retrieval (CBIR) today is an extremely active discipline. There are already review articles containing references to a large number of systems and description of the technology implemented [21,24]. A more recent review [7] reports a tremendous growth in publications on this topic. Applications of CBIR systems to medical domains already exist [17], although most of the systems currently available are based on radiological images. A query-by-example CBIR involves providing the CBIR system with an example image and retrieves the most visually similar images. This is our goal as described later.

Most of the work in dermatology has focused on skin cancer detection. Different techniques for segmentation, feature extraction and classification have been reported by several authors. Concerning segmentation, Celebi et al. [3] presented a systematic overview of recent border detection methods: clustering followed by active contours are the most popular. Numerous features have been extracted from skin images, including shape, colour, texture and border properties [26,23,14]. Classification methods range from discriminant analysis to neural networks and support vector machines [22,16,4]. These methods are mainly developed for images acquired by epiluminescence microscopy (ELM or dermoscopy) and they focus on melanoma, which is actually a rather rare, but quite dangerous, condition whereas other skin cancers are much more common.

To our knowledge, there are few CBIR systems in dermatology. Chung et al. [5] created a skin cancer database. Users can query the database by feature attribute values

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 312–319, 2010.

(shape and texture), or by synthesised image colours. It does not include a query-by-example method, as do most common CBIR systems. Their report concentrates on the description of the web-based browsing and data mining. However, nothing is said about database details (number, lesion types, acquisition technique), nor about the performance of the retrieval system. Celebi et al. [2] developed a system for retrieving skin lesion images based on shape similarity. The novelty of that system is the incorporation of human perception models in the similarity function. Results on 184 skin lesion images show significant agreement between computer assessment and human perception. However, they only focus on silhouette shape similarity and do not include many features (colour and texture) described in other papers by the same authors [4]. Rahman et al. [20] presented a CBIR system for dermatoscopic images. Their approach include image processing, segmentation, feature extraction (colour and textures) and similarity matching. Experiments on 358 images of pigmented skin lesions from three categories (benign, dysplastic nevi and melanoma) are performed. A quantitative evaluation based on the precision curve shows the effectiveness of their system to retrieve visually similar lesions (average precision $\simeq 60\%$). Dorileo et al. [9] presented a CBIR system for wound images (necrotic tissue, fibrin, granulation and mixed tissue). Features based on histogram and multispectral co-occurrence matrices are used to retrieve similar images. The performance is evaluated based on measurements of precision ($\simeq 50\%$) on a database of 215 images. All these approaches only consider a few classes of lesions and/or do not exploit many useful features in this context.

Dermatology atlases containing a large number of images are available online [8,6]. However, their searching tool only allows query by the name of the lesion. On the other hand, the possibility of retrieving images based on visual similarity would greatly benefit both the non-expert users and the dermatologists. There is a need for CBIR as a decision support tool for dermatologists in the form of a display of relevant past cases, along with proven pathology and other suitable information [17,20]. CBIR could be used to present cases that are not only similar in diagnosis, but also similar in appearance and cases with visual similarity but different diagnoses. Hence, it would be useful as a training tool for medical students and researchers to browse and search large collection of disease related illustrations using their visual attributes.

Motivated by this, we propose a CBIR approach for skin lesion images. The present work focuses on 5 common classes of skin lesions: Actinic Keratosis (AK), Basal Cell Carcinoma (BCC), Melanocytic Nevus / Mole (ML), Squamous Cell Carcinoma (SCC), Seborrhoeic Keratosis (SK). Our system mainly relies on colour and composite texture features, evolved using genetic algorithms, and gives values of precision between 67% and 82%. The use of the evolved composite features improves the precision by about 7%. The structure of the paper is as follows. Section 2 defines the simple features. Section 3 is devoted our new proposal. Section 4 defines the similarity criteria. Results are presented in 5. Conclusions follow.

2 Feature Extraction

CBIR requires the extraction of several features from each image, which, consequently, are used for computing similarity between images during the retrieval procedure. These

features describe the content of the image and that is why they must be appropriately selected according to the context. The features have to be discriminative and sufficient for the description of different pathologies. Basically, the key to attaining a successful retrieval system is to choose the right features that represent each class of images as uniquely as possible. Many feature extraction strategies have been proposed [26,23] from the perspective of classification of images as malignant or benign. Different features attempt to reflect the parameters used in medical diagnosis, such as the *ABCD* rule for melanoma detection [12]. These features are certainly effective for the classification purpose, as seen from the performance of some classification-based systems in this domain, claiming a correct classification up to 100% [16] or specificity/sensitivity of $92.34\%/93.33\%$ [4]. However, features good for classification or distinguishing one disease from another may not be suitable for retrieval and display of similar appearing lesions. In this retrieval system, we are looking for similar images in term of colour, texture, shape, etc. By extracting good representative features, we may be able to identify images similar to an unknown query image, whether it belongs to the same disease group or not. Skin lesions appear mainly characterised by their colour and texture. In this section we will describe simple features that can capture such properties. Later we will describe how to evolve composite features from these simple ones.

Colour features are represented by the mean colour $\mu = (\mu_R, \mu_G, \mu_B)$ of the lesion and their covariance matrix Σ. Let $\mu_X = \frac{1}{N}\sum_{i=1}^{N} X_i$ and $C_{XY} = \frac{1}{N}\left[\sum_{i=1}^{N} X_i Y_i\right] - \mu_X \mu_Y$, where: N is the number of pixels in the lesion, X_i the colour component of channel X $(X, Y \in \{R, G, B\})$ of pixel i. Assuming to use the original RGB (Red, Green, Blue) colour space, the covariance matrix is: $\Sigma = \begin{bmatrix} C_{RR} & C_{RG} & C_{RB} \\ C_{GR} & C_{GG} & C_{GB} \\ C_{BR} & C_{BG} & C_{BB} \end{bmatrix}$. In this work, RGB, HSV (Hue, Saturation, Value) and CIE_Lab, CIE_Lch (Munsell colour coordinate system [20]) and Otha [19] colour spaces are used. A number of normalisation techniques have been applied before extracting colour features. We normalised each colour component by the average of the same component of the healthy skin of the same patient, because it had best performance. After experimenting with the 5 different colour spaces, we choose the normalised RGB, because it gave slightly better results than the other colour spaces.

Texture features are extracted from generalised co-occurrence matrices (CGM), that are the extension of the co-occurrence matrix [11] to multispectral images. Assume an image I having N_x columns, N_y rows and N_g grey levels. Let $L_x = \{1, 2, \cdots, N_x\}$ be the columns, $L_y = \{1, 2, \cdots, N_y\}$ be the rows, and $G_x = \{0, 1, \cdots, N_g - 1\}$ be the set of quantised grey levels. Let u and v be two colour channels. The generalised co-occurrence matrices are: $P_\delta^{(u,v)}(i, j) = \#\{((k, l), (m, n)) \in (L_y \times L_x) \times (L_y \times L_x) | I_u(k, l) = i, I_v(m, n) = j\}$ i.e. the number of co-occurrences of the pair of grey level i and j which are a distance $\delta = (d, \theta)$ apart. In our work, the pixel pairs (k, l) and (m, n) have distance $d = 1, \cdots, 6$ and orientation $\theta = 0°, 45°, 90°, 135°$, i.e. $(m = k+d, n = l)$, $(m = k+d, n = l+d)$, $(m = k, n = l+d)$, $(m = k-d, n = l+d)$. In order to have orientation invariance for our set of GCMs, we averaged the matrices with respect to θ. Quantisation levels $N_G = 64, 128, 256$ are used for the three colour spaces: RGB, HSV and CIE_Lab. From each GCM we extracted 12 texture features:

energy, contrast, correlation, entropy, homogeneity, inverse difference moment, cluster shade, cluster prominence, max probability, autocorrelation, dissimilarity and variance as defined in [11], for a total of 3888 texture features (12 features $\times$ 6 inter-pixel distances $\times$ 6 colour pairs $\times$ 3 colour spaces $\times$ 3 grey level quantisations).

Texture features are also extracted from the sum- and difference-histograms (SDHs) as proposed by Unser [25]. We generalised the SDHs by considering the intra- and inter-plane sum- and difference-histograms: $h_{S,D}^{(u,v)}(i) = \#\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) | I_u(k,l) \pm I_v(m,n) = i\}$. We constructed a set of SDHs varying pixel displacement, orientation, quantisation level, and colour spaces. From each SDH we extracted 15 features: sum mean, sum variance, sum energy, sum entropy, diff mean, diff variance, diff energy, diff entropy, cluster shade, cluster prominence, contrast, homogeneity, correlation, angular second moment, entropy as defined in [25], as well as the relative illumination invariant features described by Münzenmayer [18], for a total of other 9720 features (15 features $\times$ 2 illumination invariants $\times$ 6 inter-pixel distances $\times$ 6 colour pairs $\times$ 3 colour spaces $\times$ 3 grey level quantisations).

3 Evolutionary Feature Synthesis

Evolutionary algorithms have already been applied to feature synthesis problems. Aurnhammer [1] and Lam et al. [13] described the use of genetic programming to generate texture features and reported very promising results on image classification problems. Li et al. [15] proposed a hybrid of a co-evolutionary genetic programming and expectation maximisation algorithm applied on partially labelled data. They show that their algorithm outperforms support vector machines in the sense of both the classification performance and the computational efficiency in the testing phase.

In our work, each synthesised feature is derived by combining simple features using a series of operators. A genetic algorithm (GA) [10] is used in this phase.

The main issues in applying a GA to any problem are selecting an appropriate encoding representation of the solutions, defining an adequate evaluation function (fitness), and choosing the values of the parameters used by the algorithm (e.g. population size, crossover, etc.). In the case of synthesised features there are two basic items: the index of the simple features (among the 13608 extracted) to be selected and the operators used to combine them. Each chromosome is composed of two parts: a part which encodes the index set of the simple features and a part which encodes the operators. In this work we present results obtained using 6 operators: $\{1, 2, +, -, *, /\}$. Each operator is applied to a pair of features. The first 2 operators mean that only the first or the second features of the pair is chosen. The last 4 operators perform the given mathematical operation on the two features of the pair. The fitness is the number of correctly retrieved images, i.e. the images belonging to the same class as the query image. We averaged it using each image in the database as query image, and asking the system to retrieve 10 similar images for each presented image (not retrieving itself).

In the GA, the feature indexes and the operators are encoded in the chromosomes as integer numbers. Each chromosome contains 10 features and 5 operators (one for each pair of the 10 features). The implementation of mutation and crossover on integer numbers is straightforward, with the condition to generate children satisfying the range

and integer constraints on decision variables. Other GA parameters (determined after a number of experiments varying such parameters) are: 200 individuals, 0.9 crossover rate, 0.01 mutation rate, stochastic uniform selection. The stopping criteria is upon reaching the maximum number of generations (30) or having a change in the fitness of less than 10^{-6}. Results reported later are the average over 20 runs.

4 Similarity Matching

The retrieval system is based on a similarity measure defined between the query image Q and a database image I.

For colour covariance-based features, the Bhattacharyya distance metric $D_C(Q,I) = \frac{1}{8}(\mu_Q - \mu_I)^T \left[\frac{(\Sigma_Q+\Sigma_I)}{2}\right]^{-1}(\mu_Q - \mu_I) + \frac{1}{2}\ln\frac{\left|\frac{(\Sigma_Q+\Sigma_I)}{2}\right|}{\sqrt{|\Sigma_Q||\Sigma_I|}}$ is used, where μ_Q and μ_I are the average colour (over all pixels in the lesion) feature vectors, Σ_Q and Σ_I are the covariance matrices of the lesion of Q and I respectively (computed as described in Section 2), and $|\cdot|$ denotes the matrix determinant. The Euclidean distance $D_T(Q,I) = \|f^Q_{comp} - f^I_{comp}\| = \sqrt{\sum_{i=1}^{m}(f_i^Q - f_i^I)^2}$ is used for distances between the composite features f_{comp}, evolved as previously described. where m is the number of features: $m = 5$ for the composite features, $m = 10$ for the simple features used for comparison.

We aggregated the two distances into a similarity matching function as:

$$S(Q,I) = w_C \cdot D_C(Q,I) + (1 - w_C) \cdot D_T(Q,I) \tag{1}$$

where w_C is a weighting factor that has been selected experimentally, after trying all the values: $\{0.1, 0.2, \cdots, 0.9\}$. In our case, $w_C = 0.7$ gave the best results.

5 Results and Evaluation

Our image database comprises 533 lesions, belonging to 5 classes (20 AK, 116 BCC, 224 ML, 20 SCC, 153 SK). Images are acquired using a Canon EOS 350D SRL camera, having a resolution of about 0.03 mm. Lesions are segmented using the method described in [27]. The ground truth used for the experiments is based on agreed classifications by 2 dermatologists. Feature synthesis is performed using only 100 images (20 for each class randomly chosen). The effectiveness of the proposed retrieval system is then evaluated on the entire database.

One example of the composite feature set is shown in Figure 1, together with the plot of class distribution of one of them, where it can be seen it slightly distinguishes ML and SK from AK, BCC, SCC. A typical screen-shot of our CBIR system is shown in Figure 2(a).

For medical image retrieval systems, the evaluation issue is very often neglected in most of the papers [17]. In an information retrieval scenario, *precision* is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search (*scope*), and *recall* is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents. We show average precision/scope curves obtained by evaluating top N retrieved

#	colours	feature name	dist	q.level	operator
1	BB	homogeneity ii	4	64	+
	HS	entropy	3	64	
2	ab	correlation	2	128	-
	HH	diff entropy ii	5	256	
3	aa	diff variance ii	2	64	-
	bb	entropy ii	5	128	
4	RB	inv diff moment	3	128	1
5	VV	diff energy ii	4	128	+
	La	cluster prom	3	256	

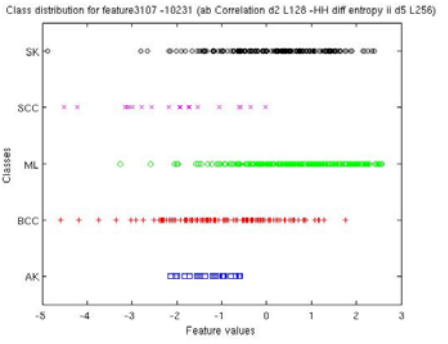
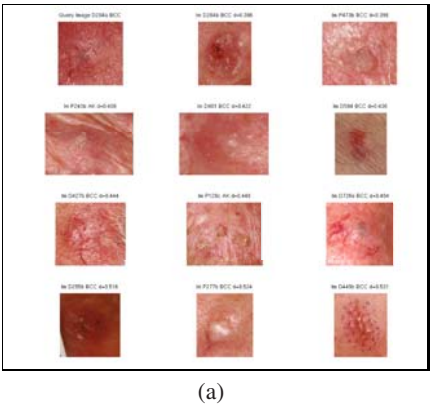


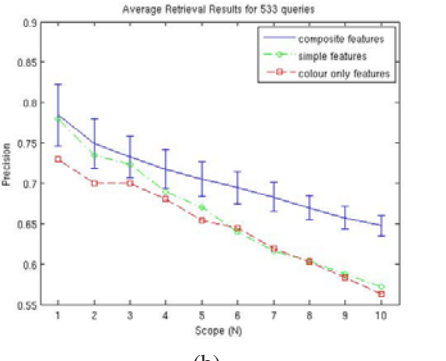
Fig. 1. Example of evolved composite features, and class distribution of feature 2

results (scope). We compare our results with the results obtained by the same system except using simple features. The simple features are selected by a GA (using the same parameters as the other GA). Similarity function (1) is used with $m = 10$ in D_T. Our 5 composite features originate from 10 simple features, therefore we decided to compare 5 composite features against 10 simple ones.

Figure 2(b) shows the precision/scope curves obtained using the composite features synthesised by our method. Precision/scope curves obtained using simple features and only colour features are shown for comparison. Note that using $m = 5$ composite features outperform $m = 10$ simple features and that the use of the composite features improves the precision by about 7%, where at scope=1 the difference is 1%. The comparison with the performances obtained using only the colour features makes clear the improvement of our system due to the composite texture features.



(a)



(b)

Fig. 2. (a) A screenshot showing retrieved images similar to the query image (top left image). (b) Precision/Scope curves using our evolved composite features, simple features and colour only features. Vertical bars report performance over 20 runs (mean $\pm$ std).

As far we know, our system is the first query-by-example CBIR system for these 5 classes of lesions, therefore comparison with other system is not possible. The use of a system developed for generic image retrieval gave very poor results on our data.

6 Conclusions

We have presented a CBIR system as a diagnostic aid for skin lesion images. We believe that presenting images with known pathology that are visually similar to an image being evaluated may provide intuitive clinical decision support to dermatologists. We have shown that the use of evolved composite features improves the performance of the system compared to the use of a larger number of standard features. Given the encouraging results obtained using a small set of feature combination operators we plan to investigate the use of a larger number of operators that combine an arbitrary number of features. Genetic programming (GP) may offer several advantages over GA. Further studies will also include the extraction of other texture-related features (i.e. fractal dimension, Gabor- and Tamura-based) as well as shape and boundary features. We plan also to include relevance feedback, which is commonly used in image retrieval, but has not yet been used for medical images.

Acknowledgements. We thank the Wellcome Trust for funding this project.

References

1. Aurnhammer, M.: Evolving texture features by genetic programming. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 351–358. Springer, Heidelberg (2007)
2. Celebi, M.E., Aslandogan, Y.A.: Content-based image retrieval incorporating models of human perception. In: International Conference on Information Technology: Coding and Computing (ITCC 2004), vol. 2, pp. 241–245. IEEE Computer Society, Los Alamitos (2004)
3. Celebi, M.E., Iyatomi, H., Schaefer, G., Stoecker, W.V.: Lesion border detection in dermoscopy images. Computerized Medical Imaging and Graphics 33(2), 148–153 (2009)
4. Celebi, M.E., Kingravi, H.A., Uddin, B., Iyatomi, H., Aslandogan, Y.A., Stoecker, W.V., Moss, R.H.: A methodological approach to the classification of dermoscopy images. Computerized Medical Imaging and Graphics 31(6), 362–373 (2007)
5. Chung, S.M., Wang, Q.: Content-based retrieval and data mining of a skin cancer image database. In: International Conference on Information Technology: Coding and Computing (ITCC 2001), pp. 611–615. IEEE Computer Society, Los Alamitos (2001)
6. Cohen, B.A., Lehmann, C.U.: Dermatlas (2000-2009), dermatology Image Altas, http://dermatlas.med.jhmi.edu/derm/
7. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Computing Surveys 40(2), 5:1–5:60 (2008)
8. Dermnet: the dermatologist's image resource (2007), dermatology Image Altas, http://www.dermnet.com/
9. Dorileo, E.A.G., Frade, M.A.C., Roselino, A.M.F., Rangayyan, R.M., Azevedo-Marques, P.M.: Color image processing and content-based image retrieval techniques for the analysis of dermatological lesions. In: 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2008), August 2008, pp. 1230–1233 (2008)

10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
11. Haralick, R.M., Shanmungam, K., Dinstein, I.: Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics 3(6), 610–621 (1973)
12. Johr, R.H.: Dermoscopy: alternative melanocytic algorithms–the ABCD rule of dermatoscopy, menzies scoring method, and 7-point checklist. Clinics in Dermatology 20(3), 240–247 (2002)
13. Lam, B., Ciesielski, V.: Discovery of human-competitive image texture feature extraction programs using genetic programming. In: Deb, K., Poli, R., Banzhaf, W., Beyer, H.G., Burke, E., Darwen, P., Dasgupta, D., Floreano, D., Foster, J., Harman, M., Holland, O., Lanzi, P.L., Spector, L., Tettamanzi, A., Thierens, D., Tyrrell, A. (eds.) GECCO 2004, Part II. LNCS, vol. 3103, pp. 1114–1125. Springer, Heidelberg (2004)
14. Lee, T.K., Claridge, E.: Predictive power of irregular border shapes for malignant melanomas. Skin Research and Technology 11(1), 1–8 (2005)
15. Li, R., Bhanu, B., Dong, A.: Feature synthesized EM algorithm for image retrieval. ACM Transaction on Multimedia Computing Communications and Applications 4(2), 10:1–10:24 (2008)
16. Maglogiannis, I., Pavlopoulos, S., Koutsouris, D.: An integrated computer supported acquisition, handling, and characterization system for pigmented skin lesions in dermatological images. IEEE Transactions on Information Technology in Biomedicine 9(1), 86–98 (2005)
17. Müller, H., Michoux, N., Bandon, D., Geissbuhler, A.: A review of content-based image retrieval systems in medical applications - clinical benefits and future directions. International Journal of Medical Informatics 73, 1–23 (2004)
18. Münzenmayer, C., Wilharm, S., Hornegger, J., Wittenberg, T.: Illumination invariant color texture analysis based on sum- and difference-histograms. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) DAGM 2005. LNCS, vol. 3663, pp. 17–24. Springer, Heidelberg (2005)
19. Ohta, Y.I., Kanade, T., Sakai, T.: Color information for region segmentation. Computer Graphics and Image Processing 13(1), 222–241 (1980)
20. Rahman, M.M., Desai, B.C., Bhattacharya, P.: Image retrieval-based decision support system for dermatoscopic images. In: IEEE Symposium on Computer-Based Medical Systems, pp. 285–290. IEEE Computer Society, Los Alamitos (2006)
21. Rui, Y., Huang, T.S., Chang, S.F.: Image retrieval: Current techniques, prominsign directions, and open issues. Journal of Visual Communication and Image Representation 10, 39–62 (1999)
22. Schmid-Saugeons, P., Guillod, J., Thiran, J.P.: Towards a computer-aided diagnosis system for pigmented skin lesions. Computerized Medical Imaging and Graphics 27, 65–78 (2003)
23. Seidenari, S., Pellacani, G., Pepe, P.: Digital videomicroscopy improves diagnostic accuracy for melanoma. Journal of the American Academy of Dermatology 39(2), 175–181 (1998)
24. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(12), 1349–1380 (2000)
25. Unser, M.: Sum and difference histograms for texture classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(1), 118–125 (1986)
26. Wollina, U., Burroni, M., Torricelli, R., Gilardi, S., Dell'Eva, G., Helm, C., Bardey, W.: Digital dermoscopy in clinical practise: a three-centre analysis. Skin Research and Technology 13, 133–142 (2007)
27. Xiang, L., Aldridge, B., Ballerini, L., Fisher, R., Rees, J.: Depth data improves skin lesion segmentation. In: Yang, G.-Z., Hawkes, D., Rueckert, D., Noble, A., Taylor, C., et al. (eds.) MICCAI 2009. LNCS, vol. 5762, pp. 1100–1107. Springer, Heidelberg (2009)

An Evolutionary Method for Model-Based Automatic Segmentation of Lower Abdomen CT Images for Radiotherapy Planning

Vitoantonio Bevilacqua[1,2], Giuseppe Mastronardi[1,2], and Alessandro Piazzolla[1]

[1] Department of Electrical and Electronics, Polytechnic of Bari,
Via Orabona, 4 – 70125 Bari – Italy
bevilacqua@poliba.it
[2] e.B.I.S. s.r.l. (electronic Business in Security), Spin-Off of Polytechnic of Bari,
Via Pavoncelli, 139 – 70125 Bari – Italy

Abstract. Segmentation of target organs and organs at risk is a fundamental task in radiotherapy treatment planning. Since its completion carried out by a radiation oncologist is really time-consuming, there is the need to perform it automatically. Unfortunately there is not a universal method capable to segment accurately every anatomical structure in every medical image, so each problem requires a study and an own solution. In this paper we analyze the problem of segmentation of bladder, prostate and rectum in lower abdomen CT images and propose a novel algorithm to solve it. It builds a statistical model of the organs analyzing a training set, generates potential solutions and chooses the segmentation result evaluating them on the basis of an aprioristic knowledge and the characteristics of patient image, using Genetic Algorithms. Out method has been tested qualitatively and quantitatively and offered good performance.

Keywords: Segmentation, lower abdomen CT, radiotherapy planning, genetic algorithms.

1 Introduction

Radiotherapy is a medical treatment which consists of delivering high speed ionizing radiation, typically X-rays, in order to cure various kinds of cancer pathologies. Radiation therapy works by disrupting DNA activity; because cancer cells have smaller ability to repair DNA damages, they are inherited through cell division and accumulate themselves leading the diseased cells to death or to a slower reproduction. During the radiation process also healthy cells could be hit, causing side effects that, in some organs, are acute or chronic. This is the reason why an important goal is to administer a high dose of radiation to tumor tissues, avoiding as much as possible normal tissues and specially organs at risk (OARs). The achievement of this aim requires an accurate treatment planning.

Nowadays treatment planning is guided by 3D medical images, acquired in several modalities like Computed Tomography (CT) and Magnetic Resonance (MR). In inverse planning a radiation oncologist delineates target volumes and organs at risk in

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 320–327, 2010.

each slice and defines doses; then, the treatment plan is formulated by an automatic algorithm. In [3] this task is performed by a genetic algorithm-based framework that computes beam intensity, beam shape and beam orientation, considering it as an optimization problem. The planning accuracy could be improved removing artifacts in patient images. They are generally due to patient motion, beam hardening and metallic object. In [2] Bevilacqua et al. presented a method based on Artificial Neural Networks to reduce metallic artifacts produced by reperi, which are used to help the radiation oncologist in localizing some structures on patient images but are removed during the treatment.

In this paper we focus on the organ segmentation. The novel model-based method of automatic segmentation we propose is based on Genetic Algorithms (GA) [7]. GAs are very useful in image understanding applications, especially in medical imaging, as summarized in survey papers [1,12], where this task becomes harder for several reasons as a considerable variability of shapes, lack of contrast, missing or diffuse boundaries and artifacts. A promising approach consists in formulating the segmentation problem as an optimization problem and using GAs to determine the parameters set which maximizes (or minimizes) a quality (or poorness) criterion, searching it into a parameter search space. The application of GAs does not guarantee that the global optimal will be found, but empirical results show that in many cases the final solution is close to it and, for this reason, they represent an efficient and robust tool on which to base the segmentation process. In facts, since in medical imaging the search space is very discontinuous and nosily, the classical gradient search techniques are often sentenced to fail; on the other hand it is too large to conduct an exhaustive search. GAs permit to avoid the main weakness of another popular class of segmentation methods for medical images based on an energy-minimizing approach, the traditional deformable models, presented by Kass et al. in [8], that evolve toward a minimal energy configuration driven by image forces but are prone to convergence to local minima, resulting in bad results where the initialization is too rough. On the other hand, an automatic initialization close to the boundaries is possible only when the inter-patient variability of anatomical structures is not strong, but this is not the case of lower abdomen CT.

There are many works in literature about medical image segmentation and GAs. They typically learn from example how to segment the target image. In [4], Cagnoni et al. employed GAs at first in order to evolve a contour detector, making use of a small set of manually traced contours acquired from the target image, and then in order to find the parameters of an elastic-contour model. In [10] shape and textural priors, derived from a dataset of images manually segmented, affect the evolution, which is conducted by a GA, of a segmenting curve represented by a level set function. In [14] GAs are used to overcome typical weak spots of traditional deformable models as model initialization, deformable parameter selection and local optima avoiding; moreover, GAs have been combined with constrained shape deformations to explore the search space.

In this paper we deal with the problem of automatic identification of prostate, bladder and rectum contours in pelvic CT. It has been faced with several means in literature but in the most of cases previous works focus on only some of the three organs or sometimes require a user intervention; some existing approaches make use of deformable surfaces [6, 15], region growing [13], genetic algorithms [10] and artificial neural

networks [11]. The method we propose employs a statistical representation of organs' features, using Point Distribution Models, obtained by Principal Component Analysis, and Probability Distributions of gray values; this knowledge has been exploited during the search of organs of interest, which is based on GAs.

Analysis of this problem, methods, materials and validations are described in the next chapters.

2 Analysis of the Problem

Organs of interest considered in this study are bladder, prostate and rectum. Analyzing the problem, we drew the following considerations:

1. boundaries between bladder and prostate are not strong, this is due to a poor contrast and makes problematic using a lot of existing techniques based on gradient operator;
2. rectum contains internal dark structures. It means that some techniques could be misled by their boundaries that are obviously different from the rectum boundaries so, for example, it becomes almost impossible to use a snake ([8]) without initializing it very close to the real edges;
3. the shapes of the same organ could be remarkably dissimilar from a patient to another;
4. intensity profile of some organs are similar, so it is inadvisable to base the research only on the gray values;
5. shapes of some organs are similar, so it's inadvisable to base the research only on the shapes.

The observations 1) and 2) indicate that, in this context, good performance could be obtained only by a robust segmentation method: for example, if we apply a local search, modifying a deformable surface placed on target image, we will run the risk of overcoming the right boundaries or of converging to false ones during its evolution. Moreover, the outstanding variability, pointed out by comment 3), discourages the employment of an atlas constituted by a single labeled image, and consequently it would be preferable to build a statistical model of the shapes. So we developed a new algorithm that makes use of the knowledge about shapes, gray values and typical pose, on the basis of the remarks 3), 4) and 5). We used a dataset of labeled images to train a statistical model of each organ, described by a Point Distribution Model (PDM); a detailed explanation about this topic is exhaustively discussed in [15]. The research process follows a top-down strategy: we generate shapes compatible with the model shape, placing them on the target image with different parameters of pose, and then measure their fitness against the vicinity of possible edge pixels and the similarity of the histograms; we entrust this task to GAs.

3 Workflow of Our Method

Our segmentation method for lower abdomen CT is set out in two phases. The first is a training phase, in which the system acquires knowledge about shapes, gray values

distribution inside organs, gray values range at the interface between the inside and the outside of the organs, average position and size. The latter is the research phase, in which, given a target image, the following tasks are performed: pose parameters initialization through a first genetic algorithm, pose parameters refinement and shape parameters searching through a second genetic algorithm and finally error correction.

Note that it is possible to set up the system once, executing only the research phase when one wants to segment an image.

3.1 The Training Phase

The training phase is aimed to build statistical models of bladder, prostate and rectum and involves shapes, gray values, positions and dimensions. It requires a dataset of images in which a radiation oncologist has segmented manually the structures; in order to obtain brighter performance, it would be better to construct it with CT images related to many different patients.

The characteristics of the shapes are learnt creating PDMs. As regard the information about gray values, exploiting the training dataset, for each organ of interest we calculate their internal distribution and the range of intensity at the boundary between the interior and the exterior of the anatomical structure. Moreover, the system grasps the average position and size of each organ.

3.2 The Research Phase

The research phase aspires to find the instance of the shape model and the pose parameters that best fit with the image target. This is done by generating admissible solutions and evaluating them considering two kinds of information extracted from the patient's image: the gray values and the corresponding edge map. The edge map is a binary image in which the pixels are marked as edge pixels or not and it could be the output provided by the Canny algorithm with an adaptive selection of the threshold values to make to come out the most part of the edge pixels of bladder, prostate and rectum and as less as possible the edge pixels relative to other structures. In particular, upper threshold is defined as the littlest gray-value such that the sum of the occurrences of littler gray-values of gaussian smoothed image, with a standard deviation of one, is greater or equal to 70% of total number of pixels. Below threshold has been set multiplying by 0.4 the upper threshold. According to the observations originated in the analysis of the problem, we expect that the edge map does not show each real edge pixel between bladder and prostate and contains some spurious edge pixels inside rectum. We can improve a bit the edge map canceling the pixels marked as edge whose gray value does not belong to the range at the interface related to the organ we want to segment that has been learnt during the training phase, through a threshold operation.

The search starts with the initialization of the pose parameters and is performed by a genetic algorithm whose chromosomes are composed by four genes: the coordinates of the center of gravity, the scaling factor and the rotation angle. The fitness function to minimize is

$$f_{GA1} = \frac{dist_{Bhattacharyya}}{\sqrt{scalingFactor}} \tag{1}$$

where $dist_{Bhattacharyya}$ is the Bhattacharyya distance between the gray values distribution learnt during the training phase and that of the average segment, having the pose parameters of the individual of the population to be evaluated. The Bhattacharyya distance between two distributions H_1 and H_2 is defined as

$$dist_{Bhattacharyya}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i)H_2(i)}}{\sqrt{\sum_i H_1(i) \sum_i H_2(i)}}} \quad (2)$$

Low scores indicate a good match, instead high scores indicate bad match; therefore a perfect match is 0, a total mismatch is 1. Note that also an accurate segment could lead to a $dist_{Bhattacharyya}$ greater than 0, especially in rectum, caused by air or by other reasons. However, the presence of air doesn't reduce dramatically algorithm performance, because it doesn't alter the whole distribution.

Note that scalingFactor is 1 when the shape size matches with the average size, estimated analyzing the training set, and its presence in f_{GA1} is aimed to avoid that a very small surface with a gray-scale distribution similar to the model one could be preferred to a surface with a little less similar intensity distribution, but having a size closer to the average size.

To avoid generating individuals surely unfit, we can vary the coordinates of the center of gravity around the average coordinates of the organ to be contoured. This is just a soft constraint because the range of excursion may still be wide.

The second GA looks for the final solution, generating new individuals; its chromosomes are composed by 4+t genes, where t is computed for each organ as suggested in [14]: the four pose parameters, which can slightly differ from those found by the first GA, and other t elements representing a set of shape parameters related to the shape models learnt, which can vary within a well-defined value range, according with the general theory of PDMs reported by [14]. The fitness function to minimize is

$$f_{GA2} = dist_{edge} \cdot (1 + \sqrt{dist_{Bhattacharyya}}) \quad (3)$$

where $dist_{edge}$ is the average distance between the landmarks of the shape generated with the pose corresponding to the individual to be evaluated and the nearest edge pixel in the edge map. Note that the value of $dist_{edge}$ is used as weight for the relevance of $dist_{Bhattacharyya}$: in this way, if it is low, a more remarkable difference between gray-values distributions will be tolerated.

In the end, the errors in estimated segmentation could be partially corrected calculating a weighted mean involving the results obtained for adjacent slices, getting better results when the resolution is finer. Figure 1 shows three examples of detected organs, in which linear interpolation has been performed on points found by the algorithm; moreover, shapes have been corrected considering the results obtained for the previous and the following slices too. Since each shape, sampled with a constant number n of points, could be seen as a 2n element vector, x, we used formula (4).

$$x_z^c = \frac{x_{z+1} + 2x_z + x_{z-1}}{4} \quad (4)$$

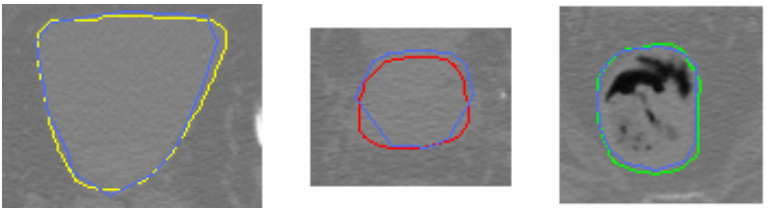


Fig. 1. Automatic segmentation (in blue) against manual segmentation of bladder (on the left), prostate (on the center), rectum (on the right)

4 Experimental Results

Our method has been implemented in C++ using the GAlib library [9] and has been tested in order to evaluate its performance. The training set we had was composed by 21 CT images (512x512 pixels) labeled manually by an only radiation oncologist, but future works will be relying on manual contouring by different experts; rectum shapes have been sampled with 16 points, instead bladder and prostate shapes have been sampled with 12 points. The segmentation has been conducted on 30 images for each organ (bladder, prostate and rectum), randomly chosen from a lower abdomen CT dataset, resulting in 90 executions of the algorithm proposed. Although GAs are stochastic methods, the algorithm has been tested once for each image, in order to reproduce a more realistic situation. Table 1 summarizes GAs parameters we used.

Table 1. GAs parameters

	GA 1	***GA 2***
Population	100	100
Generations	150	150
Crossover probability	20%	20%
Mutation probability	80%	80%
Coordinates of center of gravity (pixel)	[mean-75, mean+75]	[init.-5, init.+5]
scaling factor	[0.4, 1.6]	[init.-0.05, init.+0.05]
rotation angle (rad.)	[-0.05, 0.05]	[init.-0.05, init.+0.05]
b_i	---	$[-3\sqrt{\lambda_i}, +3\sqrt{\lambda_i}]$

The results obtained have been quantitatively measured using four surfaces overlap metrics: Target Overlap (TO), Mean Overlap (MO), Union Overlap (UO) and our Accuracy metric. Denoting with S the set of the pixels included to the segment found by our algorithm and with T the set of the pixels included to the segment manually drawn by the expert and, for this reason, considered as true, the metrics are defined as:

$$TO = \frac{|S \cap T|}{|T|} \tag{5}$$

$$MO = \frac{2|S \cap T|}{|S| + |T|} \tag{6}$$

$$UO = \frac{|S \cap T|}{|S \cup T|} \tag{7}$$

$$Accuracy = \frac{|S \cap T| - |S \setminus T|}{|T|} \tag{8}$$

Their measured values in best, worst and average case are reported in table 2.

Table 2. Metrics values

	Bladder	***Prostate***	***Rectum***
TO (best)	0.97307	0.97884	0.98271
TO (worst)	0.84818	0.71494	0.89252
TO (average case)	0.92453	0.84577	0.93350
MO (best case)	0.97223	0.91833	0.94359
MO (worst case)	0.83778	0.81630	0.88458
MO (average case)	0.92986	0.86074	0.92673
UO (best case)	0.94596	0.84900	0.89321
UO (worst case)	0.72084	0.68961	0.79305
UO (average case)	0.87053	0.75682	0.86393
Accuracy (best case)	0.94413	0.83994	0.89177
Accuracy (worst case)	0.67152	0.67821	0.74921
Accuracy (average case)	0.86026	0.72979	0.85102

Examining these results and considering that TO, MO and UO are 1 in a perfect match and 0 in a total mismatch, and that accuracy is 1 in a perfect match and a negative value in a total mismatch, we can say that the algorithm we propose has been proved to be capable of segmenting quite correctly the target CT images. Furthermore our method does not require a manual initialization of contours, so it can provide a fully automatic segmentation. The highest performance are reached in segmentation of bladder and rectum, instead are worse in prostate contouring.

5 Conclusions

In this work we have developed a novel algorithm to automatically segment organs of interest for radiotherapy in lower abdomen CT images. It exploited an a priori knowledge of the organs, taking into account their particular characteristics and the image acquisition modality, according to the observations made during the analysis of the problems: we generated a series of potential valid solutions, consistent with a statistical model, and evaluated them choosing as solution what best fits to the patient's image.

This technique could be useful to delineate automatically target organs or organs at risk during the radiotherapy planning treatment, especially to cure prostatic carcinoma; actually this task is generally performed by a radiation oncologist, consuming a lot of time. The performance of our algorithm, evaluated qualitatively and measured quantitatively using four surfaces overlap metrics, is good.

References

1. Alander, J.T.: An indexed bibliography of genetic algorithms in optics and image processing. Report series no. 94-1-OPTICS, Department of Electrical Engineering and Automation, University of Vaasa, Finland (2000)
2. Bevilacqua, V., Aulenta, A., Carioggia, E., Mastronardi, G., Menolascina, F., Simeone, G., Paradiso, A., Scarpa, A., Taurino, D.: Metallic artifacts removal in breast CT images for treatment planning in radiotherapy by means of supervised and unsupervised neural network algorithms. In: Huang, D.-S., Heutte, L., Loog, M. (eds.) ICIC 2007. LNCS, vol. 4681, pp. 1355–1363. Springer, Heidelberg (2007)
3. Bevilacqua, V., Mastronardi, G., Piscopo, G.: Evolutionary approach to inverse planning in coplanar radiotherapy. Image and Vision Computing 25(2), 196–203 (2007)
4. Cagnoni, S., Dobrzeniecki, A.B., Poli, R., Yanch, J.C.: Genetic algorithm-based interactive segmentation of 3D medical images. Im. and Vis. Comp. 17, 881–895 (1999)
5. Cootes, T.F., Hill, A., Taylor, C.J., Haslam, J.: The use of active shape models for locating structures in medical images. Im. and Vis. Comp. 12(6), 355–366 (1994)
6. Costa, M.J., Delingette, H., Novellas, S., Ayache, N.: Automatic segmentation of bladder and prostate using coupled 3D deformable models. In: Ayache, N., Ourselin, S., Maeder, A. (eds.) MICCAI 2007, Part I. LNCS, vol. 4791, pp. 252–260. Springer, Heidelberg (2007)
7. Holland, J.: Adaptation in natural and artificial systems, 2nd edn. MIT Press, Cambridge (1992)
8. Kass, M., Witkin, A.P., Terzopoulos, D.: Snakes: Active Contour Models. International Journal of Computer Vision 1(4), 321–331 (1988)
9. GAlib: A C++ library of genetic algorithms components, `http://lancet.mit.edu/ga`
10. Ghosh, P., Mitchell, M.: Segmentation of medical images using a genetic algorithm. In: Proc. of the 8th annual conference on genetic and evolutionary computation, pp. 1171–1178 (2006)
11. Lee, C.C., Chung, P.C.: Identifying abdominal organs using robust fuzzy inference model. In: IEEE int. conf. on networking, sensing and control, vol. 2, pp. 1289–1294 (2004)
12. Maulik, U.: Medical image segmentation using genetic algorithms. IEEE Transactions on Information Technology in Biomedicine 13(2), 166–173 (2009)
13. Mazonakis, M., Damilakis, J., Varveris, H., Prassopoulos, P., Gourtsoyiannis, N.: Image segmentation in treatment planning for prostate cancer using the region growing technique. The British journal of radiology 74, 243–248 (2001)
14. McIntosh, C., Hamarneh, G.: Genetic algorithm driven statistically deformed models for medical image segmentation. In: ACM Workshop on medical applications of genetic and evolu-tionary computation, in conjunction with GECCO (2006)
15. Pekar, V., McNutt, T.R., Kaus, M.R.: Automated model-based organ delineation for radiotherapy planning in prostatic region. International Journal of Radiation Oncology Biology Physics 60(3), 973–980 (2004)

Evolution of Communicating Individuals

Leonardo Bocchi[1], Sara Lapi[1], and Lucia Ballerini[2]

[1] Dept. of Electronics and Telecommunications, Univ. of Florence, Italy
leonardo.bocchi@unifi.it
[2] School of Informatics, University of Edinburgh, UK
lucia.ballerini@ed.ac.uk

Abstract. Verbal communication between individuals requires the parallel evolution of a vocal system capable of emitting different sounds and of an auditory system able to recognize each vocal pattern. In this work we present the evolution of a population of twins where the selection pressure is based on the ability of learning a communication pattern which allows verbal transmission of information. The fitness of each pair of twins (i.e. individuals having the same genotype) is based on the percentage of correct recognition of the perceived sounds. Results indicate the evolved communication system, in absence of noise, rapidly evolves and reaches almost 100% correct classifications, while, even in presence of a strong noise either in the channel, or in the sound generation parameters, the system can obtain a very good performance (approximately 80% correct classifications in the worst case).

1 Introduction

One of the most important aspects which differentiate human beings from other animal species is the larger capability of transmitting structured information using vocal sounds. This capability, indeed, allows the transmission of experience and learned information from one individual to another, and it is crucial for humanity, from a cultural and technological point of view. Transmission of structured information requires both a set of symbols (sounds) and a semantic interpretation for sequences of sounds to reconstruct the high level information which has been transmitted.

While the building of a semantic structure and of a language is mostly a cultural process (every person is able to learn any language), the capability of generating a large number of different sounds is structurally related to the shape of the vocal tract and to the capability of changing this shape, which allows modulation of the fundamental frequency (produced by vibration of vocal folds) producing a well defined shape of the spectrum of the emitted sound. This process gives rise to the production of vocal sounds, which are almost periodic sounds, characterized and discriminated from each other by the position of the formants (resonance frequencies of the vocal tract). Dynamic changes in shape of the vocal tract are used to produce most of the consonants, which are mostly non periodic sounds.

Together with the development of a vocal tract for emission of sounds, it is necessary to develop an auditory system which is able to characterize different sounds and discriminate vowels from consonants and from noise. As a first approximation, however, the auditory system of humans are not structurally different from the auditory

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 328–335, 2010.

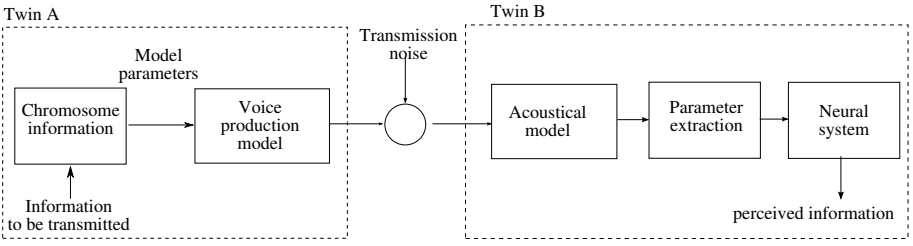


Fig. 1. Schematic representation of the evolving genotype: Twin A is emitting a sound, while Twin B is learning to recognize the emitted sound

systems of other mammalians, which in some cases presents a higher capability than the human auditory system (for instance larger and positionable ears which give a better localization of sound source). Moreover, as already noted before, communication between individuals is not only a matter of genetic evolution, but also involves a learning phase of each individual, before he can actually communicate with other individuals of the same specie.

2 Methods

In this work we emulate the evolution of a population, which is pressed from evolutionary constraints to learn to communicate. The fitness of the genotype is related to the speed and accuracy of the communication which can take place.

As communication requires the presence of similar individuals who actively try to communicate, we simulate that each genotype of the evolving population give birth to two identical twins, which try to exchange some piece of communication. The genotype, in our simulation, affect only the vocal tract of the individuals, as we suppose the auditory system in all the population shares the same functionality. This hypothesis, which allows us to greatly reduce the complexity of the algorithm, derives from the observation that in nature the basic structure of the auditory system is basically unchanged in all high level species.

The other part which takes an active (and probably, the most important) part in the development of the communications system is the learning by each individual of the proper patterns. This learning is (in humans) performed inside the brain (Broca region). In our system, the role of the brain is represented by a neural network which can be trained to listen to sounds produced by an individual having the same genotype, in order to discriminate between the different vocal emissions.

The overall communication model is represented in Figure 1. The whole system is composed of several blocks, describing the voice production part and the recognition part.

2.1 Voice Production Model

The vocal tract can be functionally described as a sound source (the vocal folds in the glottis) connected to an acoustic filter (the vocal tract) which can be modeled as a tube

of varying section [8]. Several mathematical models have been introduced to describe the voice generation process [2,3,6,7,9], but the most common representation, although with a relatively large degree of approximation, is based on a linear model:

$$P(\omega) = S(\omega)T(\omega)R(\omega) \tag{1}$$

where the voice spectrum P is related to the source spectrum S, filtered by the response frequency of the vocal tract T and by the lips radiation characteristics R. In this hypothesis, each of the three components can be independently modeled.

The source can be modeled either as a white noise source, which is rather unrealistic but gives good results for consonant simulation, or as a pulse modulated airflow source, which represents the opening and closing cycles of the vocal folds as a train of almost triangular pulses. In our model, for simplicity, the source is simulated as a train of unitary triangular pulses with a repetition frequency f_0, randomly selected in the interval from 50Hz to 100Hz, similar to the physiological range of human voices.

The model of the vocal tract is usually represented as a lossless tube, which is further simplified by assuming the tube is composed of a series of N cylindrical sections. Using this assumption, the airflow inside the tube can be analyzed using mono dimensional equations. If we consider a cylindrical section of the tube (of index k, $1 \leq k \leq N$), having transversal area equal to A_k and length l_k, the relation between the air speed u_k and the air pressure p_k inside the tube, at time t and position x, can be written as:

$$p_k(x,t) = \frac{\rho c}{A_k}\left[u_k^+(t - x/c) + u_k^-(t - x/c)\right] \tag{2}$$

$$u_k(x,t) = u_k^+(t - x/c) - u_k^-(t - x/c) \tag{3}$$

where c is the propagation speed, ρ is the air density, and u_k^+ and u_k^- represent the waves traveling in the positive and negative direction inside the tube. At each connection point between two consecutive sections of the tube, each wave is partly transmitted and partly reflected. The reflection coefficient r_k can be evaluated by using the continuity of the flow over the junction [13]:

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \tag{4}$$

Assuming a constant value of l_k across all sections, the time delay for each wave is constant, and the transfer function of a section can be described, in the z domain and in matrix notation, as:

$$\mathbf{U}_k = \mathbf{Q}_k \mathbf{U}_{k+1}, \tag{5}$$

where

$$\mathbf{U}_k = \begin{bmatrix} U_k^+(z) \\ U_k^-(z) \end{bmatrix} \text{ and } \mathbf{Q}_k = \begin{bmatrix} \frac{z^{1/2}}{1+r_k} & \frac{-r_k z^{1/2}}{1+r_k} \\ \frac{-r_k z^{1/2}}{1+r_k} & \frac{z^{1/2}}{1+r_k} \end{bmatrix} \tag{6}$$

From this equation, it can be determined [10] the overall transfer function:

$$V(z) = \frac{0.5 * z^{-N/2} * (1 + r_0) * \prod_{k=1}^{N}(1 + r_k)}{D_k(z)} \tag{7}$$

where r_0 is the reflection coefficient at the junction between glottis and the vocal tract and $D_k(z)$ is a polynomial:

$$D(z) = [1, -r_0] \begin{bmatrix} 1 & -r_1 \\ -r_1 z^{-1} & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1 & -r_N \\ -r_N z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{8}$$

The resulting transfer function is therefore represented as an all-poles lattice filter, with coefficients which depend on the values of transversal areas A_k.

The resulting signal $P(\omega)$ has been evaluated, in the time domain, as the convolution of the source impulse train and the frequency response of the filter. Before transmission, the signal has been normalized by dividing it by its variance. To remove transitory effects, we generated a pulse train composed on N=2048 samples and transmitted only the second half of the signal, obtaining a waveform including 1024 samples.

2.2 Acoustic Model

The acoustic model of the auditory system is based on the extraction of the MFCC (Mel-Frequency Cepstral Coefficients) vector [11]. The algorithm used to evaluate the MFCC can be described as follow:

- Split the input signal $y(t)$ in frames $y_n(t)$ of equal length, usually a power of 2 to ease following steps.
- Evaluate the power spectrum $P_y(\omega)$ of the frame, using short term Fourier transform, applying an adequate windowing function $w(t)$ (we used Hanning window):

$$P_y(\omega) = \left|\mathcal{F}(y_n(t)w(t))\right|^2 \tag{9}$$

- Calculate the energy S_k in each of the Mel windows

$$S_k = \sum_{\omega} W_k(\omega) S(\omega) \tag{10}$$

  where $1 \leq k \leq 13$, and $W_k(\omega)$ is the triangular weighting function centered on the k-th Mel window in the Mel scale.
- The DCT (Discrete Cosine Transform) of the logarithm of the values S_k is computed [5] to obtain the desired coefficients c_m:

$$c_m = \sum_{k=1}^{M} log(S_k) \cos\left[m(k - 0.5)\frac{\pi}{M}\right] \tag{11}$$

  where $1 \leq m \leq 13$, which is the selected order of the MFCC.

2.3 Interpretation Model

Once each frame of the received audio signal has been associated to the MFCC vector, a neural network is used to identify the signal as one of the possible vowel sounds. The neural network is a standard feed forward network, having an input layer of 13 units (MFCC coefficients), a hidden layer of 10 units, and an output layer corresponding to the number of possible vowels which can be emitted by the speaker. Network is trained, during each fitness evaluation, using the backpropagation algorithm with adaptive learning rate.

2.4 Genotype

In Figure 1, a generic genotype gives birth to two identical twins, A and B, which are trying to learn to communicate. Because of the symmetry of the situations, we suppose the twin A can produce a vocal sound selected from the set $S = S_1, S_2, \ldots, S_N$, each element of which represents an information which can be selected from the set $I = I_1, I_2, \ldots, I_N$. The genetic information stored in the chromosomes of the individual describes the features of the vocal tract, encoding the capability of the individual to produce different sounds. The chromosome, therefore, stores the transverse section of the vocal tract for each of the emitted sounds. The number of vocal sounds which the individual is able to generate has been assumed fixed and equal to five. The number of different sections has been fixed equal to ten, which gives a reasonable number of degrees of freedom, keeping the computational resources at an acceptable level. The number of sections which are required to generate a set of artificial sounds which are almost not discriminable from pure vocal sounds is considerably higher (up to forty, in some works [1, 4, 12]). Therefore, the chromosome is constituted by a vector of 50 float numbers, representing the ten cross-sectional areas for each of the five possible sounds.

2.5 Learning Phase

The task of twin B is to listen to the sound $\hat{S}_i$, which can be corrupted by a random noise, and to correctly identify the information I_i which was intended to be transmitted by twin A. At birth, similar to a human baby, twin B is unable to understand the information and therefore a learning phase is required. In this phase, twin A "teaches" twin B by repeating a series of known informations I_i, which twin B listens to. From the numerical point of view, this requires the production of training set composed of a given number (30) of samples of each vocal sound associated to each information I_i. This set is then used, together with the known values of I_i, to train the neural network representing the "brain" of twin B.

2.6 Fitness Evaluation

Once the training phase is completed, it is possible to evaluate the communication capability, by generating an unknown information which needs to be transmitted from A to B. The evaluation of the fitness of the individual is related to the number of correct identifications over a set of vocal sounds which are produced by the speaking twin, and received through the transmission channel. A test set, including 30 sound samples for each of the different sounds (for a total of 150 samples) is generated by twin A. The sound is transmitted (adding noise) to twin B, which listens to and classifies each sample in one of the five classes, using the trained neural network.

To allow a good selection of the individuals, the fitness function f has been assumed equal to the ratio (CC) of correctly identified frames (i.e. short segments) of the received samples: $f = CC$. Therefore, the optimal individual, which correctly identifies all frames of the received sounds, has a fitness values equal to one, while an individual which fails to identify any frame has a fitness value of zero.

2.7 Reproduction and Selection

The structure of the chromosome allows the use of standard crossover and mutation (Gaussian) operators. The only particular care which is required is to limit the area function to be strictly positive (to this end we introduced a minimal value ϵ of the area to avoid numerical instabilities of the filtering stage).

The selection of the population is based on the stochastic uniform method. This method lays out a line in which each parent corresponds to a section of the line of length proportional to its scaled value. The algorithm moves along the line in steps of equal size. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.

3 Results

The system has been tested in three different environments: in the ideal case, where no noise contamination is present, and the signal is received exactly as it was generated. In the second experiment, we added a white transmission noise, while in the last experiment, we also introduced a generation error, where the actual transverse sections used to generate each sound are affected by a random error with respect to the values specified in the chromosome.

In absence of noise, the different sounds appear to be easily discriminated from each other, and evolution, with a large enough population size, reaches almost optimal performance (more than 99% correct identification, in all runs of the algorithm) in a few tens of generations.

In the second experiment, we used a noisy channel, where random white noise is added to the transmitted signal. We simulated different noise levels, up to a variance of the noise $\sigma = 0.33$, which means approximately 30% of the signal variance. Simulation results indicate the evolved system is able to mantain a good level of reliability (almost 100% correct classifications) also in case of very high noise levels. Even with a noise level equal to the 50% of the signal variance, a population of 30 individuals and 100 generations, the best individuals are able to classify approximately 95% of the received sounds. A further increase of the noise shows that at a noise level of 100% (variance of the noise equal to the variance of the signal), the evolved population achieves lower performances, as shown in the example plot reported in Figure 2. In this case, the evolution is unable to obtain the optimal individual, and best performance is about 90% of correct classifications. Even in this last case, the performances (both as average and as best fitness) reach their minimum value approximately after 70-80 generations.

A second possible noise source has been identified in the variability of the speaker. We emulate this variability as a random error which is applied to the area vector each time the individuals emit a sound. This error represents, therefore, the error which the individual commits in shaping it vocal tract as specified in the chromosome. Errors occurring in each section are independent of each other, with a Gaussian distribution with zero mean and unitary variance.

Although this source of error may potentially have a larger impact on the frequency response of the vocal tract, during the evolution the performance of individuals are only marginally affected by the noise. Figure 3, top, reports the typical evolution trend

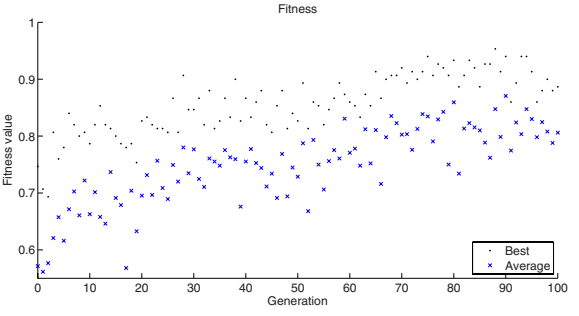


Fig. 2. Evolution of the fitness in case of additive white noise on the channel, noise level 100%

obtained with a noise level equal to the 20% of the signal variance, a population size of 20 individuals and a number of generations equal to 200. On the average, the best individuals are able to classify approximately 95% of the received sounds. It can be observed that, in this situation the fitness values keep improving almost until the end of the simulation.

Increasing the noise level to 33%, the evolved population achieves sensibly lower performances, as shown in the example plot reported in Figure 3, bottom. In this case, the evolution is unable to obtain the optimal individual, stagnates before 100 generations, and best performance, on average, is about 80% of correct classifications.

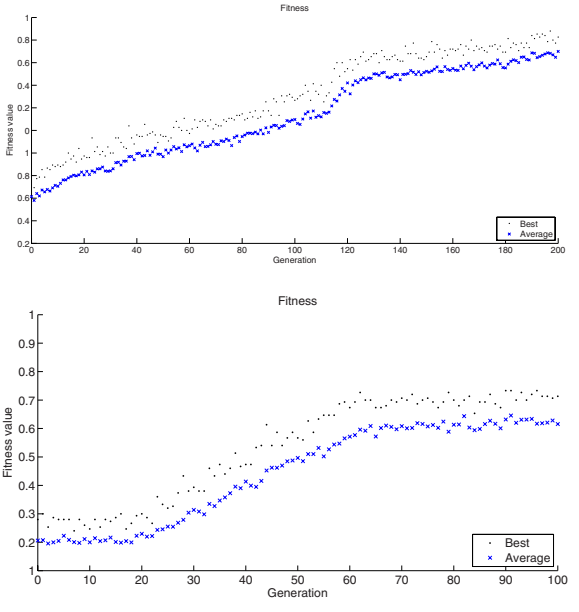


Fig. 3. Evolution of the fitness in case of additive white noise on the channel and on the vocal tract shape, levels 20% (top) and 33% (bottom)

4 Conclusions

In this paper, a simulation of the evolution of voice communication has been described. Results indicate that during the evolution the individuals can achieve a high reliability in the transmission of the information through an acoustic channel, also in presence of high levels of additive noise. The model appears to be quite robust also against errors in the shaping of the vocal tract.

Future development aims to introduce a more refined model of the chromosome representation which introduces physical constraints on the shape of the vocal tract, as well as the possibility to evolve a variable number of sounds, in order to better emulate the different number of vowels in the various spoken languages.

Acknowledgements. We thank Steven McDonagh for proof reading the manuscript.

References

1. Beautemps, D., Badin, P., Laboissière, R.: Deriving vocal-tract area functions from midsagittal profiles and formant frequencies: A new model for vowels and fricative consonants based on experimental data. Speech Communication 16(1), 27–47 (1995)
2. Bonada, J., Loscos, A., Cano, P., Serra, X., Kenmochi, H.: Spectral approach to the modeling of the singing voice. In: Proceedings of 111th AES Convention (2001)
3. Christophe, B.D., Henrich, N.: The voice source as a causal/anticausal linear filter. In: Proc. ISCA ITRW VOQUAL 2003, pp. 15–19 (2003)
4. Clément, P., Hans, S., Hartl, D., Maeda, S., Vaissière, J., Brasnu, D.: Vocal tract area function for vowels using three-dimensional magnetic resonance imaging. a preliminary study. Journal of Voice 21, 522–530 (2007)
5. Deller, J.J.R., Hansen, J.H.L., Proakis, J.G.: Discrete-Time Processing of Speech Signals, 2nd edn. Wiley-IEEE Press, Chichester (1999)
6. Kob, M., Alhäuser, N., Reiter, U.: Time-domain model of the singing voice (1999)
7. Macon, M.W., Clements, M.A.: Sinusoidal modeling and modification of unvoiced speech. IEEE Transactions on Speech and Audio Processing, 557–560 (1997)
8. Maeda, S.: A digital simulation method of the vocal-tract system. Speech Communication 1, 199–229 (1982)
9. Narayanan, S., Alwan, A.: Noise source models for fricative consonants. IEEE Transactions on Speech and Audio Processing 8, 2000 (2000)
10. Rabiner, L., Schafer, R.: Digital Processing of Speech Signals. Prentice Hall, Englewood Cliffs (1978)
11. Rabiner, L.R., Juang, B.H.: Fundamentals of Speech Recognition. Prentice Hall, Englewood Cliffs (1993)
12. Story, B., Titze, I.: Parameterization of vocal tract area functions by empirical orthogonal modes. J. Phonetics 26, 223–260 (1998)
13. Wakita, H.: Direct estimation of the vocal tract shape by inverse filtering of acoustic speech waveforms. IEEE Transactions on Audio and Electroacoustics 21(5), 417–427 (1973)

Dynamic Data Clustering Using Stochastic Approximation Driven Multi-Dimensional Particle Swarm Optimization

Serkan Kiranyaz[1], Turker Ince[2], and Moncef Gabbouj[1,*]

[1] Tampere University of Technology, Tampere, Finland
{serkan.kiranyaz,moncef.gabbouj}@tut.fi
[2] Izmir University of Economics, Izmir, Turkey
turker.ince@ieu.edu.tr

Abstract. With an ever-growing attention Particle Swarm Optimization (PSO) has found many application areas for many challenging optimization problems. It is, however, a known fact that PSO has a severe drawback in the update of its global best (*gbest*) particle, which has a crucial role of guiding the rest of the swarm. In this paper, we propose two efficient solutions to remedy this problem using a stochastic approximation (SA) technique. For this purpose we use simultaneous perturbation stochastic approximation (SPSA), which is applied only to the *gbest* (not to the entire swarm) for a low-cost solution. Since the problem of poor *gbest* update persists in the recently proposed extension of PSO, called multi-dimensional PSO (MD-PSO), two distinct SA approaches are then integrated into MD-PSO and tested over a set of unsupervised data clustering applications. Experimental results show that the proposed approaches significantly improved the quality of the MD-PSO clustering as measured by a validity index function. Furthermore, the proposed approaches are generic as they can be used with other PSO variants and applicable to a wide range of problems.

Keywords: Particle Swarm Optimization, stochastic approximation, multi-dimensional search, gradient descent, dynamic data clustering.

1 Introduction

The particle swarm optimization (PSO) [4,10,11] exhibits certain similarities with the other evolutionary algorithms (EAs) [2]. The common point of all is that EAs are in population based nature and they can avoid being trapped in a local optimum. Thus they can find the optimum solutions; however, this is never guaranteed. In a PSO process, a swarm of particles (or agents), each of which represent a potential solution to an optimization problem, navigate through the search (or solution) space. One major drawback of PSO is the direct link of the information flow between particles and the global-best particle, *gbest*, which primarily "guides" the rest of the swarm and thus resulting in the creation of similar particles with some loss of diversity. Hence this phenomenon increases the probability of being trapped in local optima [7] and it is the main cause of the premature convergence problem especially when the search

* This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence Program (2006 - 2011).

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 336–343, 2010.

space is of high dimensions [10] and the problem to be optimized is multi-modal [7]. This makes it clear that at any iteration of a PSO process, *gbest* is the most important particle; however, it has the poorest update equation, i.e. when a particle becomes *gbest*, it resides on its personal best position (*pbest*) and thus both social and cognitive components are nullified in the velocity update equation. Although it guides the swarm during the following iterations, ironically it lacks the necessary guidance to do so effectively. In that, if gbest is (likely to get) trapped in a local optimum, so the rest of the swarm due to the aforementioned direct link of information flow. This deficiency has been raised in a recent work [5] where an artificial GB particle, the *aGB*, is created at each iteration as an alternative to *gbest*. However, the underlying mechanism for creating the *aGB* particle, the so-called fractional GB formation (FGBF), is not generic, rather problem dependent.

For the problem of finding a root θ^* (either minimum or maximum point) of the gradient equation: $g(\theta) \equiv \frac{\partial L(\theta)}{\partial \theta} = 0$ for some differentiable function $L : R^p \to R^1$, when g is present and L is a uni-modal function, there are powerful deterministic methods for finding the global θ^* such as traditional steepest descent and Newton-Raphson methods. However, in many real problems g cannot be observed directly and/or L is in multi-modal nature, which presents many deceiving local optima. This brought the era of the stochastic optimization algorithms, which can estimate the gradient and may avoid being trapped into a local optimum due to their stochastic nature. One of the most popular stochastic optimization techniques is stochastic approximation (SA), in particular the form that is called "gradient free" SA. Among many SA variants the one and somewhat different SA application is called simultaneous perturbation SA (SPSA) proposed by Spall in [8].

In this paper we shall propose two approaches, one of which drives *gbest* efficiently or simply put, *guides* it with respect to the function (or error surface). The idea behind this is quite simple: since the velocity update equation of *gbest* is quite poor, SPSA as a simple yet powerful search technique is used to *drive* it instead. The second approach has a similar motivation with the FGBF proposed in [5], i.e., an artificial Global Best (*aGB*) particle is created by SPSA this time, which is applied over the personal best (*pbest*) position of the *gbest* particle. The *aGB* particle will then guide the swarm instead of *gbest* if and only if it achieves a better fitness score than the (personal best position of) *gbest*. Note that both approaches *only* deal with the *gbest* particle and hence the internal PSO process remains as is. They are then applied to the multi-dimensional extension of PSO, the MD-PSO technique proposed in [5], which can find the optimum dimension of the solution space. SA-driven (SAD) MD-PSO is then tested and evaluated against the standalone MD-PSO application over several data clustering problems.

2 Proposed Technique: SAD MD-PSO

2.1 SPSA Overview

The goal of the deterministic optimization methods is to minimize a loss function $L : R^p \to R^1$, which is a differentiable function of θ and the minimum (or maximum) point θ^* corresponds to zero-gradient point, i.e.

$$g(\theta) \equiv \left.\frac{\partial L(\theta)}{\partial \theta}\right|_{\theta=\theta^*} = 0 \tag{1}$$

As mentioned earlier, in cases where more than one point satisfies this equation (e.g. a multi-modal problem), then such algorithms may only converge to a local minimum. Moreover, in many practical problems, g is not readily available. This makes the SA algorithms quite popular and they are in the general SA form:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k) \tag{2}$$

where $\hat{g}_k(\hat{\theta}_k)$ is the estimate of the gradient $g(\theta)$ at iteration k and a_k is a scalar gain sequence satisfying certain conditions [8]. There are two common SA methods: finite difference SA (FDSA) and simultaneous perturbation SA (SPSA). FDSA adopts the traditional Kiefer-Wolfowitz approach to approximate gradient vectors as a vector of p partial derivatives where p is the dimension of the loss function. On the other hand, SPSA has all elements of $\hat{\theta}_k$ perturbed simultaneously using only two measurements of the loss function as,

$$\hat{g}_k(\hat{\theta}_k) = \frac{L(\hat{\theta}_k + c_k\Delta_k) - L(\hat{\theta}_k - c_k\Delta_k)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ . \\ . \\ . \\ \Delta_{kp}^{-1} \end{bmatrix} \tag{3}$$

where the p-dimensional random variable Δ_k is usually chosen as Bernoulli ± 1 distribution and c_k is a scalar gain sequence satisfying certain conditions [8]. Spall [8] presents conditions for convergence of SPSA (i.e. $\hat{\theta}_k \to \theta^*$) and show that under certain conditions both SPSA and FDSA have the same convergence ability –yet SPSA needs only 2 measurements whereas FDSA needs 2p. This makes SPSA our natural choice for driving gbest in both approaches. SPSA has 5 parameters. Spall [9] recommended to use values for A (the stability constant), α, and γ as 60, 0.602 and 0.101, respectively. However, he also concluded that "*the choice of both gain sequences is critical to the performance of the SPSA as with all stochastic optimization algorithms and the choice of their respective algorithm coefficients*". This especially makes the choice of gain parameters *a* and *c* critical for a particular problem. i.e. Maryak and Chin [6] varied them with respect to the problem whilst keeping the other three (A, α, and γ) as recommended.

2.2 SAD MD-PSO

As mentioned earlier, in this work two distinct SAD MD-PSO approaches are proposed, each of which is only applied in each dimension on the *gbest* of the positional PSO whilst keeping the internal PSO processes (both positional and dimensional)

intact. Since both SPSA and positional PSO are iterative processes, in both approaches SPSA can thus easily be integrated into PSO by using the same iteration count (i.e. $t \equiv k$). The following sub-sections will detail each approach.

A1) First SAD MD-PSO approach: gbest update by SPSA

In this approach, at each iteration *gbest* particle is updated using SPSA. This requires the adaptation of the SPSA elements (parameters and variables) and integration of the internal SPSA part (within the loop) appropriately into the PSO pseudo-code. Due to space limitations, we have to skip the pseudo code details of this approach. Note that such a "plug-in" approach will not change the internal PSO structure and only affects the *gbest* particle's movement. It only costs two extra function evaluations and hence at each iteration the total number of evaluations is increased from *S* to *S*+2 (recall that *S* is the swarm size).

Since the fitness of each particle's current position is computed within the PSO process, it is possible to further diminish this cost to only *one* extra fitness evaluation per iteration. Let $\hat{\theta}_k + c_k\Delta_k = xx_a^d(t)$ and thus $L(\hat{\theta}_k + c_k\Delta_k)$ is known *a priori*. Then naturally, $\hat{\theta}_k - c_k\Delta_k = xx_a^d(t) - 2c_k\Delta_k$, which is the only (new) location where the (extra) fitness evaluation ($L(\hat{\theta}_k - c_k\Delta_k)$) has to be computed. Once the gradient ($\hat{g}_k(\hat{\theta}_k)$) is estimated, then the next (updated) location of the *gbest* will be: $xx_a^d(t+1) = \hat{\theta}_{k+1}$. Note that the difference of this "low-cost" SPSA update is that $xx_a^d(t+1)$ is updated (estimated) *not* from $xx_a^d(t)$, instead from $xx_a^d(t) - c_k\Delta_k$. Note that in a MD-PSO process there is a distinct *gbest* particle, *gbest(d)*. So SPSA is applied individually over the position of each *gbest(d)* if it (re-) visits the dimension d, (i.e. $d = xd_{gbest}(t)$). Therefore, there can be $2(D_{\max} - D_{\min})$ number of function evaluations, indicating a significant cost especially if a wide dimensional range is used. However, this is a theoretical limit, which can only happen if $gbest(i) \neq gbest(j)$ *for* $\forall i, j \in [D_{\min}, D_{\max}], i \neq j$ and all particles altogether visit the particular dimensions in which they are gbest (i.e. $xd_{gbest(d)}(t) = d, \forall t \in [1, iterNo]$). Especially in a wide dimensional range, note that this is highly unlikely due to the dimensional velocity, which makes particles move (jump) from one dimension to another at each iteration. It is straightforward to see that under the assumption of a uniform distribution for particles' movements over all dimensions within the dimensional range, SAD MD-PSO too, would have the same cost overhead as the SAD PSO. Experimental results indicate that the practical overhead cost is only slightly higher than this.

A2) Second SAD PSO approach: aGB formation by SPSA

The second approach replaces the FGBF operation proposed in [5] with the SPSA to create an *aGB* particle. SPSA is basically applied over the *pbest* position of the *gbest* particle. The *aGB* particle will then guide the swarm instead of *gbest* if and only if it achieves a better fitness score than the (personal best position of) *gbest*. SAD MD-PSO pseudo-code as given in Table 1 can then be plugged into the MD-PSO.

Note that in this approach, there are three extra fitness evaluations (as opposed to two in the first one) at each iteration. Yet as in the first approach, it is possible to further diminish the cost by *one* (from three to two fitness evaluations per iteration). In order to create an *aGB* particle for all dimensions in the given range (i.e. $\forall d \in [D_{\min}, D_{\max}]$) SPSA is applied individually over the personal best position of each *gbest(d)* particle and furthermore, the aforementioned competitive selection ensures that $xy^d_{aGB}(t), \forall d \in [D_{\min}, D_{\max}]$ is set to the best of the $xx^d_{aGB}(t+1)$ and $xy^d_{aGB}(t)$. As a result, the SPSA creates one *aGB* particle providing (potential) GB solutions ($xy^d_{aGB}(t+1), \forall d \in [D_{\min}, D_{\max}]$) for all dimensions in the given dimension range. The pseudo-code of the second approach as given in Table 1 can then be plugged in between steps 3.2 and 3.3 of the MD-PSO pseudo-code, given in [5].

Table 1. MD-PSO Plug-in for the second approach at dimension *d*

A2) *SAD MD-PSO Plug-in* (ξ **, *a, c, A,*** α , γ)

1. Create a new *aGB* particle, $\{xx^d_{aGB}(t+1),\ xy^d_{aGB}(t+1)\}$ *for* $\forall d \in [D_{\min}, D_{\max}]$
2. For $\forall d \in [D_{\min}, D_{\max}]$ do:
   - 2.1. *Let k=t,* $\hat{\theta}_k = x\hat{y}^d(t)$ *and L=f*
   - 2.2. *Let* $a_k = a/(A+k)^\alpha$ *and* $c_k = c/k^\gamma$
   - 2.3. *Compute* $L(\hat{\theta}_k + c_k\Delta_k)$ *and* $L(\hat{\theta}_k - c_k\Delta_k)$
   - 2.4. *Compute* $\hat{g}_k(\hat{\theta}_k)$ *using Eq. (3)*
   - 2.5. *Compute* $xx^d_{aGB}(t+1) = \hat{\theta}_{k+1}$ *using Eq. (2)*
   - 2.6. If ($f(xx^d_{aGB}(t+1)) < f(xy^d_{aGB}(t))$) then $xy^d_{aGB}(t+1) = xx^d_{aGB}(t+1)$
   - 2.7. Else $xy^d_{aGB}(t+1) = xy^d_{aGB}(t)$
   - 2.8. If ($f(xy^d_{aGB}(t+1)) < f(xy^d_{gbest(d)}(t))$ then $xy^d_{gbest(d)}(t) = xy^d_{aGB}(t+1)$
3. End For.
4. Re-assign *dbest*: $dbest = \arg\min_{d \in [D_{\min}, D_{\max}]} (f(xy^d_{gbest(d)}(t))$

3 Experimental Results

In order to test each approach of the proposed SAD MD-PSO technique over clustering, we created 8 synthetic data spaces as shown in Figure 1 where white dots (pixels) represent data points. For illustration purposes each data space is formed in 2D; however, clusters are formed with different shapes, densities, sizes and inter-cluster distances to test the robustness of clustering application of the proposed approaches against such variations. Furthermore, recall that the number of clusters determines the (true) dimension of the solution space in a PSO application and hence it is also kept

varying among data spaces to test the converging accuracy to the true (solution space) dimension. As a result, significantly varying complexity levels are established among all data spaces to perform a general-purpose evaluation of each approach.

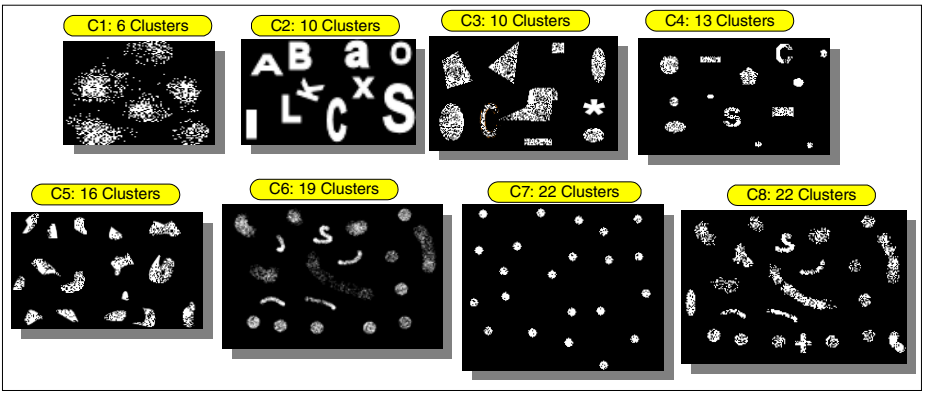


Fig. 1. 2D synthetic data spaces carrying different clustering schemes

The maximum number of iterations is set to 10000 and the use of cut-off error as a termination criterion is avoided since it is not feasible to set a unique ε_C value for all clustering schemes. For MD-PSO, we used the swarm size, *S*=200 and for both SAD MD-PSO approaches, a reduced number is used in order to ensure the same number of evaluation among all competing techniques. *w* is linearly decreased from 0.75 to 0.2 and we again used the recommended values for *A*, α, and γ as 60, 0.602 and 0.101, whereas *a* and c are set to 0.4 and 10, respectively. For each dataset, 20 clustering runs are performed.

For visual evaluation, Figure 2 presents the *worst* and the *best* clustering results of the two competing techniques, standalone vs. SAD MD-PSO, based on the highest (worst) and lowest (best) fitness scores achieved among the 20 runs. In the figure each cluster is represented in one of the three color codes (red, green and blue) for illustration purposes and each cluster centroid (each dimensional component of the *gbest* particle) is shown with a white '**+**'.The clustering results of the best performing SAD MD-PSO approach are shown whilst excluding C1 since results of all techniques are quite close for this data space due to its simplicity. Note first of all that the results of the (standalone) MD-PSO deteriorate severely as the complexity and/or the number of clusters increases. Particularly in the *worst* results, the critical errors such as under-clustering often occur with dislocated cluster centroids. For instance 4 out of 20 runs for C6 results in severe under-clustering with 3 clusters, similar to the one shown in the figure whereas this goes up to 10 out of 20 runs for C8. Although the clusters are the simplest in shape and in density for C7, due to the high solution space dimension (e.g. number of clusters = 22), even the *best* MD-PSO run is not immune to under-clustering errors. In some of the *worst* SAD MD-PSO runs too, one or few under-clusterings do occur; however, they are minority cases in general and definitely not as severe as in MD-PSO runs. It is quite evident from the *worst* and the *best* results in

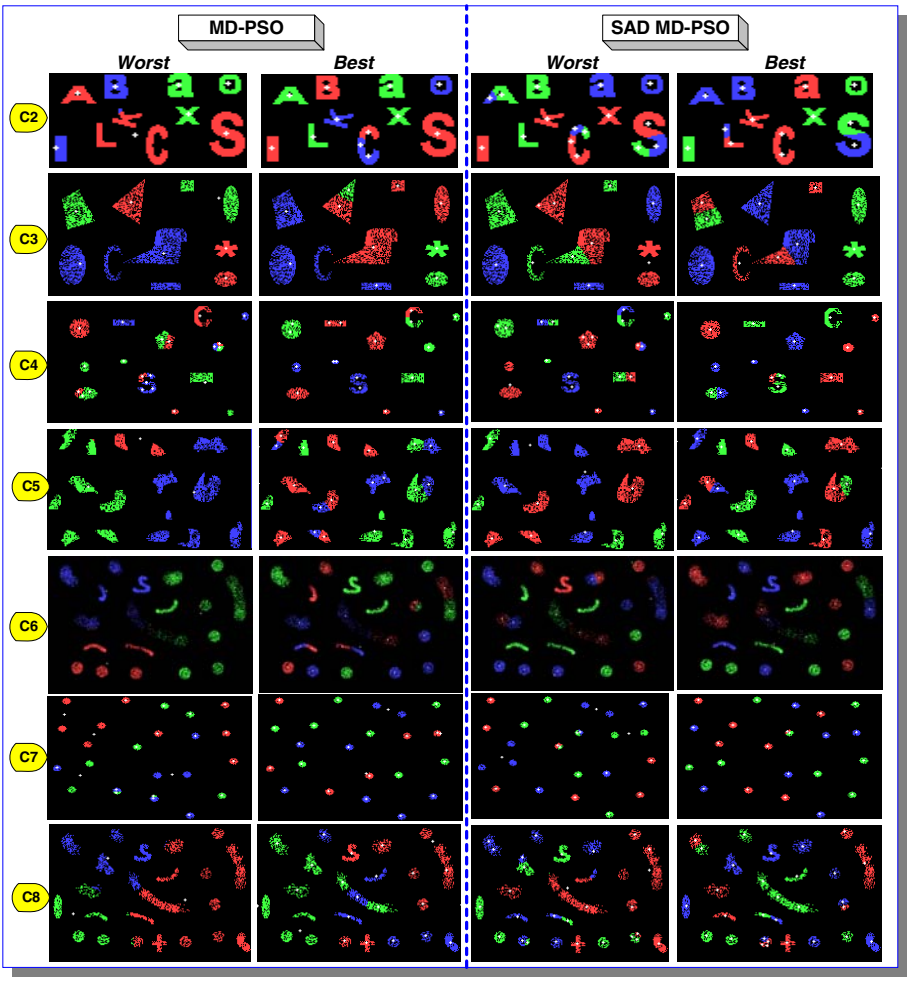


Fig. 2. The worst and the best clustering results using standalone (left) and SAD (right) MD-PSO

the figure that SAD MD-PSO achieves a significantly superior clustering quality and usually converges to a close vicinity of the global optimum solution.

4 Conclusions

In this paper, we draw the focus on a major drawback of the PSO algorithm: the poor *gbest* update. This can be a severe problem, which may cause pre-mature convergence to local optima since *gbest* as the common term in the update equation of all particles, is the primary *guide* of the swarm. SPSA is purposefully adapted to guide (or *drive)* the *gbest* particle (with simultaneous perturbation) towards the "*right*" direction with the gradient estimate of the underlying surface (or function) whilst avoiding local traps due to its stochastic nature. In that, the proposed SAD MD-PSO is not a new

MD-PSO variant or extension, rather a "guided MD-PSO" algorithm, which has an identical process with the MD-PSO as guidance is only provided to *gbest* particle –not the whole swarm.

SAD MD-PSO is then applied to the unsupervised clustering problem within which the (clustering) complexity can be thought of as synonymous to (function) modality and tested over 8 synthetic data spaces in 2D with ground truth clusters. The statistical results obtained from the clustering runs approve the superiority of SAD MD-PSO in terms of global convergence. We have applied a *fixed* set of SPSA parameters and observed that if the setting of the critical parameters, e.g. a and c is appropriate, then a significant performance gain can be achieved by SAD MD-PSO. If not, SAD MD-PSO still outperforms the standalone MD-PSO. This shows that SPSA, even without proper parameter setting still performs *better* than the PSO's native *gbest* update. Furthermore, we have noticed that the performance gap widens especially when the clustering complexity increases since the performance of the standalone MD-PSO operation, without any proper guidance, severely deteriorates. One observation worth mentioning is that the second approach on SAD MD-PSO has a significant overhead cost, which is anyway balanced by using reduced number of particles in the experiments; therefore, the low-cost mode should be used with a limited dimensional range for those applications with high computational complexity.

References

1. Abraham, A., Das, S., Roy, S.: Swarm Intelligence Algorithms for Data Clustering. In: Soft Computing for Knowledge Discovery and Data Mining book, Part IV, October 25, pp. 279–313 (2007)
2. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithm for parameter optimization. Evolutionary Computation 1, 1–23 (1993)
3. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On Cluster Validation Techniques. Journal of Intelligent Information Systems 17(2, 3), 107–145 (2001)
4. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of IEEE Int. Conf. on Neural Networks, Perth, Australia, vol. 4, pp. 1942–1948 (1995)
5. Kiranyaz, S., Ince, T., Yildirim, A., Gabbouj, M.: Fractional Particle Swarm Optimization in Multi-Dimensional Search Space. IEEE Trans. on Systems, Man, and Cybernetics (2009) (in print)
6. Maryak, J.L., Chin, D.C.: Global random optimization by simultaneous perturbation stochastic approximation. In: Proc. of the 33rd Conf. on Winter Simulation, Washington, DC, December 9-12, pp. 307–312 (2001)
7. Riget, J., Vesterstrom, J.S.: A Diversity-Guided Particle Swarm Optimizer - The ARPSO, Technical report, Department of Computer Science, University of Aarhus (2002)
8. Spall, J.C.: Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. IEEE Transactions on Automatic Control 37, 332–341 (1992)
9. Spall, J.C.: Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Trans. on Aerospace and Electronic Systems 34, 817–823 (1998)
10. Van den Bergh, F.: An Analysis of Particle Swarm Optimizers, PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa (2002)
11. Yan, Y., Osadciw, L.A.: Density estimation using a new dimension adaptive particle swarm optimization algorithm. Journal of Swarm Intelligence 3(4) (2009)

Automatic Synthesis of Associative Memories through Genetic Programming: A First Co-evolutionary Approach

Juan Villegas-Cortez[1], Gustavo Olague[2], Carlos Aviles[1], Humberto Sossa[3], and Andres Ferreyra[1]

[1] Universidad Autónoma Metropolitana - Azcapotzalco. Departamento de Electrónica. Av. San Pablo 180 Col. Reynosa, 02200. México D.F., México
jvillegas@gmail.com
[2] Centro de Investiación Científica y de Educación Superior de Ensenada CICESE. Ensenada, B.C. México
olague@cicese.mx
[3] Centro de Investigacion en Computacion CIC-IPN. México D.F., México
hsossa@cic.ipn.mx

Abstract. Associative Memories (AMs) are mathematical structures specially designed to associate input patterns with output patterns within a single stage. Since the last fifty years all reported AMs have been manually designed. The paper describes a Genetic Programming based methodology able to create a process for the automatic synthesis of AMs. It paves a new area of research that permits for the first time to propose new AMs for solving specific problems. In order to test our methodology we study the application of AMs for real value patterns. The results illustrate that it is possible to automatically generate AMs that achieve good recall performance for problems commonly used in pattern recognition research.

1 Introduction

An associative memory (AM) is a special type of Artificial Neural Network (ANN) designed to recall output patterns in terms of input patterns by means of simple operations using reduced computing structures. The enormous simplicity of AMs constitutes a very useful advantage versus traditional ANN models which have a higher structural complexity. In the former, the information is stored by a learning process where the association between input, X, and output Y, patterns is denoted as (X^k, Y^k); where, k is the corresponding association. The associative memory M is represented by a matrix whose components can be seen as the synapses of a simple neural network. The matrix M is generated from an *a priori* set of finite known associations, named as the fundamental set of associations and is represented by $\{(X^k, Y^k)|k = 1, ..., p\}$, where p is the number of associations. If $(X^k = Y^k)\forall k = 1, \ldots, p$, then M is considered as auto-associative, otherwise as hetero-associative. Also, a distorted version of a pattern X to be restored will be denoted as $\widetilde{X}$. Thus, if M is fed with a distorted version of X^k

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 344–351, 2010.

and the output being obtained is exactly Y^k; then, the recalling feature is considered as perfect. As for the associations of M, these are conformed by simple operations such as: addition, multiplication, maximum, minimum and others. In order to preserve its simplicity the process of creating new structures for AMs turns out to be a matter of science and art. Several models of AMs have been developed during the last few years, refer to:[7]. However, those AMs have several limitations; for example: their limited storage capacity, their difficulty to deal with more than one type of patterns (binary, polar, integer or real valued), their lack of robustness to deal with different kinds of noises (additive, subtractive, mixed, Gaussian, etc.); and above all, most of them work only for the purpose for which they were specially designed and not for any other. Since every model has been manually developed by the human mind, their implementation takes at least one or two years.

In this paper we improve our methodology presented in a previous report [8], inspired from biological evolution, which is based on Genetic Programming (GP). GP is based on Neodarwinian evolution principles aiming to automatically create computer programs by means of natural selection and is used in the solution of real world complex problems. The implementation of GP is commonly performed by a set of individuals or computer programs written as tree structures. These are easily evaluated in a recursive way using prefix notation; thus, every node has one function operator and every terminal has a corresponding element. It is widely acknowledged that GP has been successfully applied in a huge range of areas. In particular, we can mention those related with computer vision (see:[2],[4]). This work describes for the first time a GP system that is able to synthesize new AMs.

This article is organized as follows. Section 2 provides a brief description of the problem. In section 3 we present our co-evolutionary GP methodology for the automatic synthesis of AM through GP. Section 4 shows our results. Finally, conclusions and suggestions for further research are presented in section 5.

2 Automatic Synthesis of AMs through GP

In general, the generation of AMs with evolutionary algorithms consists of the following two-stage rule. The evolution starts as a simple ANN with a pre-established topology evolved from more than one connection [9]. The process continues with the evolution of architectures with the aim of generating different topological structures; this involves the definition of the connectivity for every synapse.

Based on [8] we introduced a very important modification using the co-evolutionary framework. Initially, we considered as a basis for the definition of an AM one simple association, using elementary arithmetic operators and having special care with vector multiplication. Then, we focused on the connectivity of every synapse that was carried-out between the components of the corresponding input and output vectors; hence, using a one-to-one correspondence.

On the whole, the design of AMs uses different sets of operators: one for association and another for the recalling stage. So, we decided to implement the co-evolutionary framework through the definition of two search spaces in three

steps. First, we defined the operators for the association stage; later, we defined the recalling space using an association matrix generated previously; and finally, we implemented the co-evolutionary approach that solved the problem using both search spaces. Here, the cooperative co-evolutionary paradigm was central to achieve our purpose of evolving AMs [3],[5].

3 The Design of AMs Using a Co-evolutionary Model

Our proposal for a GP-development of AMs comprises two populations for each evolutionary process, the solutions for the association phase are in the first population, and the solutions for the recalling phase are in the second one. The cooperative co-evolution is inspired by the ecological relationship between different species living together in the same environment, where each one provides one part of the solution for the community survival skill. In our problem these species are considered as possible solutions for each part of the global solution of our problem. In Fig. 1 we present a framework of the application of this idea. There, the solutions of AMs come from two search spaces, and the interaction between the first evolution process implied in the second evolution stage is depicted. The fitness of the whole process tries to be representative of the framework.

The components of our new model are defined through the following points (see, Fig. 2):

- Op_k^a. The evolutionary operator for the pattern association. It corresponds to the encoded genotype. The local association matrix μ_i is made by the Op_k^a operator-rule, as it is depicted in Fig. 2.
- M_k. The k-th association matrix which is taken as the sum of all local associations (μ_i), it provides the cumulative knowledge inspired by the perceptron principle.
- $T_a = \{x_j, y_j\}$. The *Terminal Set* for the association stage, $x_j \in X$ and $y_j \in Y$.
- $F_a = \{+, -, min, max, *\}$. The *Function Set* for the association stage. These functions have been defined considering the solutions reported in the literature in order to target the space of possible structures. This aims to produce individuals similar in performance to the reviewed AMs models.
- Op_p^r. The evolutionary operator-rule for the pattern recall. It comprises the input vector X_j, as well as the association matrix M_k.
- $T_r = \{v, R_1, R_2, \ldots, R_m, M_k\}$. The *Terminal Set* for the recalling stage, with $v \in X$, as the input vector, and R_i as the ith-row-vector $\in M_k$.
- $F_r = \{+, -, min, max, mytimesm\}$. The *Function Set* for the recalling stage. $mytimesm$ is defined as the multiplication operator between vector components, $mytimesm(X, Y) = [x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n]$. When the association matrix is considered, $mytimesm(X, M_k) = X * M_k$. Hence, it satisfies the dimensionality of the multiplication operator between matrices.
- $\hat{Y}_j$. The approximated Y_j pattern resulting from the application obtained with the rule Op_p^r in the recalling stage.

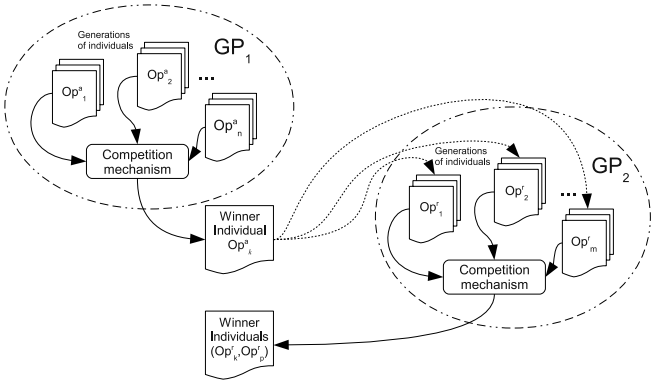


Fig. 1. Proposed framework of the cooperative co-evolution process

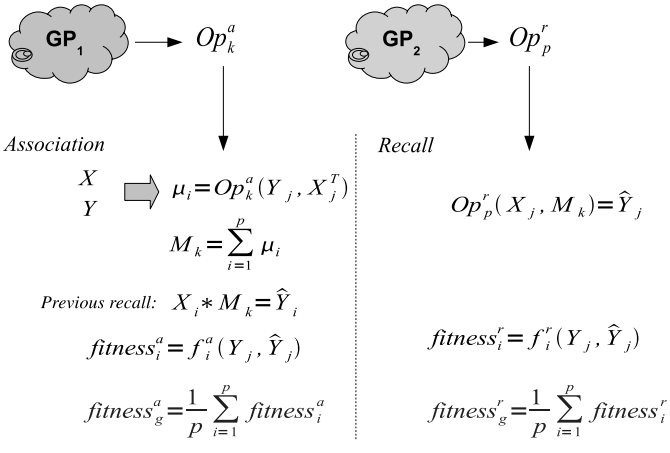


Fig. 2. Proposed co-evolutionary model for the GP-development of AMs

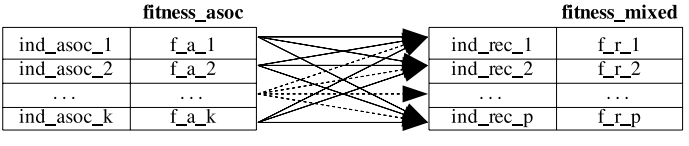


Fig. 3. Fitness model for the GP-development of AMs

The evolutionary process demands the assignation of a *fitness* value to those individuals included in both stages: association and recalling. For the sake of simplicity the *fitness function* was considered to be equal to the cosine similarity of the angle between the goal (Y) and the calculated pattern ($\widehat{Y}$), which is defined as follows:

$$f = \frac{Y \cdot \widehat{Y}}{\sqrt{Y \cdot Y}\sqrt{\widehat{Y} \cdot \widehat{Y}}} \tag{1}$$

The optimum is found when $f = 1$, corresponding to the matching of all values. The worst case takes place for $f = 0$, implying that not a single value is matched. We apply Eq. (1) as our fitness function per every stage. The evaluation was carried out in three steps per every prong. First, we applied the *association* between X and Y implemented by Op_k^a. Second, we performed a simple recalling by the known *multiplication* operator in order to have a local estimator for this step. Third, the computation of Eq. (1) between Y and $\widehat{Y}$, for all the local associations, was carried out in order to have a global fitness for the association stage $fitness_g^a$ and it was associated to the Op_k^a operator. We applied the same method during the *recalling* process. First we computed the recalled patterns using the evolutionary operator Op_r^p, this step involved the association matrix, M_k, generated with the association operator Op_a^k. Second, we computed the fitness for Op_r^p using the same fitness function of the recalling stage, the local recalling fitness, $fitness_i^r$, associated to the evolutionary operator functioning as the recalling rule. Third, a global fitness was evaluated and assigned for every operator pair (Op_a^r, Op_r^p); we took the highest-fitness grade of each solution pair. This process is shown schematically in Fig. 2 and Fig. 3.

3.1 GP Setup

The experiments were implemented using Matlab with GPLab toolbox [6]. Due to the simplicity of the function set we performed several batches of *"N"* runs, each consisting of 50 generations, each with 70 individuals for the association phase, and another *"N"* batch of 50 generations, each with 70 individuals for the recalling phase; during the last phase each evolutionated individual, resulting from the previous phase, cooperates with its own association-rule and its own fitness value in all the generations obtained during the recalling phase (see Fig. 1). After the development of all of this batch process we obtained one possible evolutionary solution for our problem (a pair of GP trees, one individual for association and its corresponding individual for recalling).

The GP parameters used in our experiments was similar to those suggested by Koza [1], taking values of 0.7 for the crossover rate and 0.3 for the mutation rate operations, respectively. Mutation was based on the *ramped-half-and-half* initialization method, which was also used to initialize the population.

4 Results and Analysis

We applied this methodology in autoassociative relationship in order to have new AMs to be applied to very-well known databases from the University of California Machine Learning Repository: (i) the Iris Plant (4 features, 3 classes, 150 instances); and (ii) the Wine database (13 features, 2 clases, 178 instances).

Results comming from the Iris Plant database are discussed as follows. The three resulting AMs suitable for this problem are shown in operators of Eq. 2

Table 1. Winner pairs of the association rules with their corresponding recalling rules, and their global fitness

Association rule	Recalling rule	Global Fitness
$Competition(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_1^r	1
$Competition(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_2^r	1
$Competition(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_3^r	1

and 3, three possible association rules for the association stage, and three for the recalling stage, then the co-evolutionary process was executed to find the winner pairs. Each individual has a tag per every evolutionary process, and the global fitness for every winner dupla is shown in Table 1.

According to Table 1, the evolutionary process produces miscellaneous solutions. Complex (Op_1^a, Op_3^r), intermediate (Op_1^a, Op_2^r), and simple (Op_1^a, Op_1^r). However, the only valid operator for the association stage turned out to be Op_1^a; see first column, Table 1. It is interesting to note that the association operators Op_2^a and Op_3^a do not generate any valid association, for their generated asociation matrices were not adequate (their combination with the recalling operator yielded poor results).

An important feature revealded through this methodology is the characteristics of the main synapses or connections playing in competition. For this example the winner rule for the recalling stage shows the rows numbers 1, 2 and 4, belonging to their association matrix M_k, generated by the association rule $O_{p_1}^a$. These rows are the most significant during the recalling phase. They are related to three features: the sepal length, the sepal width and the petal width; all of them evaluated in cm. So, the new AM shows only three of the four features as the most relevant from the Iris Plant database.

On the other hand, the results comming from the Wine database are discussed as follows. One resulting AM is shown in Eq. 4. There, the simplicity of the recalling operator compensates the complexity of the association rule. One important feature of this pattern set is revealed in the recalling rule operator. Contrary to the Iris case, where the attibute set was reduced into three relevant characteristics, the Wine case comprises all the whole features of the database. So, our methodology shows that all attributes are relevant for the association purposes to this pattern set.

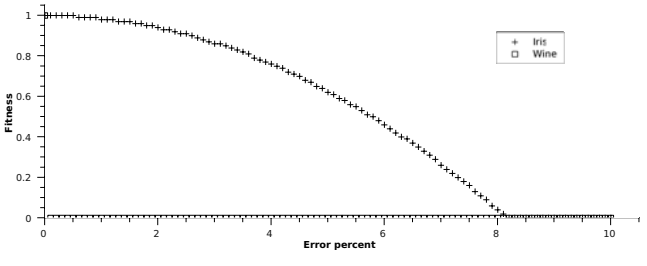


Fig. 4. Robustness to noisy patterns for the evolutionary AMs

We tested these new AMs —the solution of the Iris case (Op_1^a, Op_3^r); and the pair corresponding to Eq. 4 (Op_1^a, Op_2^r)—by adding random noise to the input pattern set X, from 0 to 10 %. The resulting fitness is depicted in Fig. 4. We observe that the recalling rate decreases slowly as the noise increases for the Iris case, on the contrary for the Wine case, which evolutionary AM works only for the training pattern set.

$$
\begin{aligned}
Op_1^a &= min(+(*(+(min(-(x,y),y),x),y),-(+(*(max(x,*(y,*(x,y))),x),\\
&\quad -(x,y)),max(+(+(max(y,x),+(y,x)),x),max(-(+(y,x),x),y)))),\\
&\quad min(+(-(min(+(y,y),y),x),+(-(+(y,x),y),y)),y))\\
Op_2^a &= min[y,(x*x-x)*min(max(x*x-x,y),y*x+y)-max(x*x-x,y)]\\
Op_3^a &= min(max(-(*(max(y,*(y,min(y,x))),min(max(-(*(y,min(y,x)),y),\\
&\quad -(-(*(*(y,-(x,+(*(y,x),y))),*(y,x)),y),-(-(x,+(+(x,x),y)),\\
&\quad +(-(y,*(y,y)),x)))),x)),y),-(-(*(*(+(y,*(x,y)),-(-(*(*(+(y,*(x,y)),\\
&\quad -(x,+(*(y,x),y))),*(y,x)),y),y)),*(y,x)),y),y)),y)
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
Op_1^r &= min[R_3+max[mytimesm(X,M_k)+min(mytimesm(X,M_k),R_1),\\
&\quad min(max[min(X,R_2),R1*X],\\
&\quad mytimesm(X,M_k))],X-[R_2+mytimesm(X,M_k)+mytimesm(X,M_k),\\
&\quad mytimesm(X,M_k)-(R_2-max(X,R_1))]+max(R_4,R_2*X)\\
&\quad -mytimesm(X,M_k)]+max[min[min(max[min(mytimesm(X,M_k),\\
&\quad R_4),R_1*X],mytimesm(X,M_k)),mytimesm(X,M_k)],X]\\
Op_2^r &= min[min(min(X,mytimesm(X,M_k)),R_1),min(mytimesm(X,M_k),\\
&\quad min(R_2,mytimesm(X,M_k)))]\\
Op_3^r &= min(X,R_3)
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
Op_1^a =& min(-(max(-(y,x),-(*(+(-(-(max(-(y,x),min(-(x,x),+(*(*(min(-(x,x),\\
&\quad +(*(-(-(-(x,y),y),min(y,min(y,x))),-(min(y,x),y)),x)),x),\\
&\quad -(min(y,x),y)),x))),y),y),x),x),+(min(*(y,x),max(-(y,x),\\
&\quad min(+(y,y),y))),-(+(x,x),y)))),+(*(+(y,x),-(min(y,x),y)),x)),y)\\
Op_2^r =& min(+(x,x),mytimesm(x,M_k))
\end{aligned}
\tag{4}
$$

5 Conclusions

We improved our first approach where we originally proposed the GP methodology for the synthesis of AMs development [8]. We obtained several AMs that fit perfectly each pattern set by means of GP coevolutionary methodology. Instead of the traditional approach that takes several years of research performed by human experts, the time necessary to generate such solutions varies from hours to days, depending on the computational time being necessary to synthesize ten

or more models. Our methodology has one very important advantage, the possibility to develop AMs specially designed for specific pattern sets. The resulting evolutionary AMs are produced by the efficiency of the GP that searches for optimized solutions. In particular, our methodology can be implemented to have several solutions per pattern applications with lower computing time. Right now we are working on improving our methodology with the use of new fitness functions and metric spaces according to the kind of the pattern sets. We expect to deal soon with challenging real world applications.

Acknowledgments

The authors give thanks to CIC-IPN and COFAA-IPN for the economical support. Authors also thank SIP-IPN under grant 20091421. This document was prepared with economical support of the European Community, the European Union and CONACYT under grant FONCICYT 93829. The content of this document is an exclusive responsibility of the IPN, CICESE and UAM, and cannot be considered as the position of the European Community.

References

1. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
2. Olague, G., Puente, C.: Honeybees as an intelligent based approach for 3d reconstruction. In: International Conference on Pattern Recognition, Hong Kong, China, August 20-24 (2006)
3. Paredis, J.: Coevolutionary computation. Artif. Life 2(4), 355–375 (1995)
4. Perez, C., Olague, G.: Learning invariant region descriptor operators with genetic programming and the f-measure. In: International Conference on Pattern Recognition (2008)
5. Potter, M.A., Jong, K.A.D.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8, 1–29 (2000)
6. Silva, S., Almeida, J.: Gplab-a genetic programming toolbox for matlab (2004), http://gplab.sourceforge.net/
7. Vázquez, R.A., Sossa, H.: Hetero-associative memories for voice signal and image processing. In: Iberoamerican Congress on Pattern Recognition, pp. 659–666 (2008)
8. Villegas-Cortez, J., Sossa, H., Aviles-Cruz, C., Olague, G.: Automatic synthesis of associative memories by genetic programming, a first approach. Research in Computing Science. Advances in Computer Science and Engineering 42, 91–102 (2009)
9. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE, 1423–1447 (1999)

A Comparative Study between Genetic Algorithm and Genetic Programming Based Gait Generation Methods for Quadruped Robots

Kisung Seo and Soohwan Hyun

Dept. of Electronic Engineering, Seokyeong University, Seoul, Korea

Abstract. Planning gaits for legged robots is a challenging task that requires optimizing parameters in a highly irregular and multidimensional space. Two gait generation methods using GA (Genetic Algorithm), GP (genetic programming) are compared to develop fast locomotion for a quadruped robot. GA-based approaches seek to optimize a pre-selected set of parameters which include locus of paw and stance parameters of initial position. A GP-based technique is an effective way to generate a few joint trajectories instead of the locus of paw positions and many stance parameters. Optimizations for two proposed methods are executed and analyzed using a Webots simulation of the quadruped robot built by Bioloid. Furthermore, simulation results for the two proposed methods are tested in a real quadruped robot, and the performance and motion features of GA-, GP -based methods are compared.

Keywords: Robot Automatic Gait Generation, Quadruped Robot, Genetic Algorithm, Joint Space Trajectory, Genetic Programming.

1 Introduction

The mobility of a walking robot including quadruped robots is distinguished from that of a wheeled robot by the ability to traverse uneven and unstructured environments [1,14]. Planning gaits for quadruped robots is a challenging task, because there are many degrees of freedom and, therefore, parameters, to be set properly [3,7]. Existing automatic generation methods for quadruped gaits include: GA-(Genetic Algorithm-) based approaches, GP- (Genetic Programming-) based approaches.

Most current GA-based approaches seek to optimize a pre-selected set of parameters. They typically belong to three groups: locus of paw, initial position, and number of movement points per cycle [3,4,6,10,12,15].

An efficient approach was proposed [13] by authors to use genetic programming to optimize joint trajectories instead of the locus of paw positions in Cartesian space. The joint-space-oriented method has to optimize only 4-6 joint trajectories, rather than the more numerous parameters of Cartesian space. GP-based search is an effective way to generate the joint trajectories in an open-ended manner.

However, it is difficult to choose a specific gait generation method for a given robot, because there are many different conditions to that influence the decision, including robot features and characteristics of problem spaces. Moreover, there is no published

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 352–360, 2010.

research comparing these three approaches in the same environments for the same robots. Therefore, experimental comparisons and analysis for automatic gait generation methods will be a useful guideline for selection of gait generation method.

In this paper, both representative gait generation methods named above are compared for the quadruped robot represented by the Bioloid Kit [11]. In order to compare the two methods in as fair and unbiased a fashion as possible, the same experimental environments and performance indexes are selected for all.

Forward velocity of gait is widely used as a performance index in previous works [3,4,7,10], which rewards forward motion and penalizes sideways diversion. Therefore, it is very reasonable to compare the two gait generation methods above according to speed of walking without sideways diversion.

In this paper, two evolutionary based and gait generation approaches are compared for the quadruped robot. To investigate the characteristics of the two approaches, implementations and experiments on GA-, GP-based gait evolution are executed for the Bioloid quadruped robot in the Webots environment. Section 2 discusses problems of gait generation and models of a quadruped robot. Section 3 describes GA-based evolution of gait in the Cartesian space. and Section 4 explains GP-based evolution of gait in the joint trajectory framework. Section 5 presents experimental results for evolved gaits, and Section 6 concludes the paper.

2 GA-Based Gait Generation in Cartesian Space

With the parameterizations described in Section 2, the problem of optimizing the gait speed becomes a parameter optimization problem in multi-dimensional space. As a first method, we use an evolutionary approach based on genetic algorithms to optimizing the locus of the robot's paw for gait generation. A conventional SGA (Simple Genetic Algorithm) [5] is chosen.

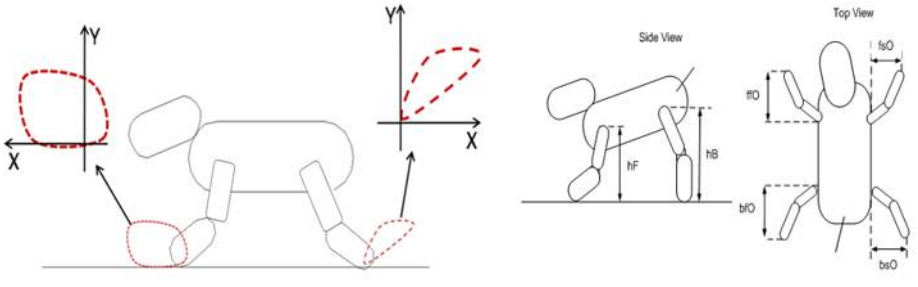


Fig. 1. An example of third-order spline locus and stance parameters for quadruped robot

To evolve the locus of paw positions for a quadruped robot, in this paper, the shape of the locus is represented by a third-order spline, which has been found by others [4] to be a preferred representation.

Not only must we select a certain type of locus, but we must also search for the optimal parameters for the given locus, such as height and width. There also exist

independent parameters for initial position (so-called "stance" parameters) as shown in Figure 1.

3 GP-Based Gait Control in Joint Space

Given that a GA-based approach utilized a large set of parameters for gait and stance, we propose an efficient approach with fewer parameters to use genetic programming to optimize joint trajectories instead of the locus of paw positions in Cartesian space. The joint-space-oriented method has to optimize only 4-6 joint trajectories, rather than the more numerous parameters of Cartesian space. GP-based search is an effective way to generate the joint trajectories in an open-ended manner.

Similar research on automatic generation of control programs for walking robots using GP exists [2], but it is focused on generating a general-purpose control program for a virtual and morphology-independent robot consisting only of simple blocks. Because there is no morphology-related information such as specified lengths or distances, it does not take into account many practical problems of a specific, real robot and it is not suitable for the optimization of real gaits. The proposed GP-based gait generation method is more practical and problem specific and deals with many real phenomena and problems presented by real robots.

The proposed approach has the following unique features. First, a solution for gait is represented by each joint's trajectory rather than by the locus of paw positions and stance parameters. Second, genetic programming is used to evolve trajectories, rather than a GA-based approach, providing flexibility in the form of trajectories being searched. Third, inverse kinematics calculations are not necessary to compute the gait of the quadruped robot. Fourth, a solution for gait is a form of continuous curve, so no interpolation process is required.

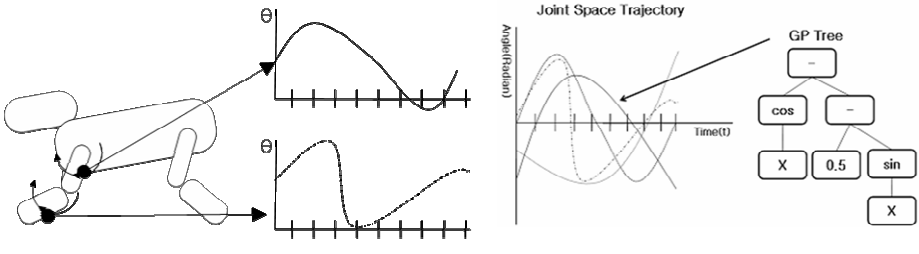


Fig. 2. Gait generation in joint space and Representation of trajectory of a joint via a GP tree

The concept of the gait generation in joint space is shown in Figure 2. The joint trajectories of shoulder and knee for Aibo are represented in 2-D space; the vertical axis is joint angle and the horizontal axis is time. Without the need for conversion of paw position from Cartesian space to a set of joint angles, a gait is determined directly by a series of joint positions (or angles), which corresponds to one cycle of paw locus in Cartesian space.

Specification of gait as a set of joint trajectories is done by evolving a polynomial function of time for each joint as a separate GP tree, but evolving them simultaneously.

The numerical expressions generated by each GP tree resemble those generated when using GP to perform symbolic regression. In Figure 2, the GP tree on the right side represents some polynomial expression that translates as shown on the left into the joint angle for one of the quadruped robot joints.

4 Experiments and Analysis

Here the above two representative gait generation methods are compared for a Bioloid Kit quadruped robot. In order to compare the two methods fairly and with as little bias as possible, the same experimental environments, performance indexes and available computational efforts are used.

Forward velocity of gait is selected as a performance index, which promotes moving straight ahead, and reduces sideways diversion. To maximize velocity during walking without falling down is good and well-defined performance index, also useful for investigating the characteristics of the search space and capabilities of each method.

4.1 Simulation Environments

The Webots [8] 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. Figure 3 shows real and simulation model of the quadruped robot by Bioloid Kit [11].

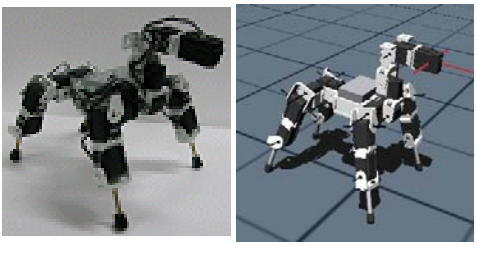


Fig. 3. Real and simulation model of the quadruped robot

4.2 Fitness Function and GA, GP Parameters

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 described in equation (1), where x is total forward distance reached, z is sideway diversion.

$$fitness = ((0.9 \times (-x)) - |(0.4 \times z)|)^2 \quad (1)$$

The GA and GP parameters specified were as shown below (Table 1).

Table 1. The GA and GP Parameters

GA Parameters	GP Parameters
Number of generations: 100	Terminal Set : Random Constants, X
Population sizes: 150	Function Set : SIN, COS, +, -, *, /
Selection: Roulette Wheel	Number of generations : 100
Crossover: Arithmetic Crossover, 0.9	Population sizes : 30*5 multiple populations
Mutation: Random Mutation, 0.1	Initial population : half_and_half
	Initial depth : 1-6
	Max depth : 15
	Selection : Tournament (size=7)
	Crossover : 0.6
	Mutation : 0.1
	Reproduction : 0.3

4.3 Simulation Results

Two gait generation methods—GA (Genetic Algorithm), GP (Genetic Programming)—are executed using Webots simulation. The number of evaluations to be used by the two methods is set to be equal, to make a fair comparison. Fifteen experiments are executed for each method, and of those fifteen, ten runs are chosen for use in summarizing statistics, eliminating those that do not work properly during replay of the simulation even though they fared well in the evolution process. These results that failed to replay properly are removed. These experiments were run on a single Core 2 Duo 2.13GHz PC with 2GB RAM.

4.3.1 Performance of Velocity

The tabular results of average and max velocities for generated gaits by GA, displayed versus number of movement points per cycle, are provided in Table 2.

The average velocity is an average of 10 iterations for the max velocity in each experiment. The maximum velocity represents the best values among the 10 experiments. “Movement points” represents the number of segments associated with the selected locus of the paw.

Table 2. Results by variation of movement points per cycle(GA)

Movement points	Average velocity	Maximum velocity
40	15.62	20.14
50	19.25	21.84
60	16.73	18.60

In the GA-based experiment, the best performance is 21.84 cm/s for max velocity (velocity) and 19.25 cm/s for average velocity of 10 experiments using 50 movement points. The max velocity with 40 movement points is better than with 60 points, but the average velocity with 40 movement points is worse than with 60 points.

The tabular results of the GP-based methods displayed against number of movement points per cycle are provided in Table 3. Here, movement points represent the number of segments associated with the obtained trajectory of the joints. The max velocity of best performance is 26.53 cm/s for 40 movement points and the average velocity is 15.04 cm/s. Lower numbers (40) of movement points show better results than higher numbers 50 and 60. Compared to the results from the GA, the GP results show more deviation between max and average.

Table 3. Results by variation of movement points per cycle(GP)

Movement points	Average velocity	Maximum velocity
40	15.04	26.53
50	13.53	18.53
60	11.28	18.01

Summarizing Tables 2 to 3, we observe that the GP-based method yielded the best max velocity, and the GA results showed superior average velocity to the other methods. The GP results showed larger deviation even though the max velocity was highest.

4.3.2 Movement Data for Gait Behavior

Tables 4-5 provides the vertical variation and average height of shoulder from ground to investigate to allow examining the motions of a quadruped robot for comparison of GA and GP. The vertical variation means the difference between the highest and lowest y coordinate of the shoulder during walking. The average height and SD represent the mean y coordinate of the shoulder during walking, measured every 32ms, and its standard deviation. The unit is cm. Both values are averages of 10 experiments.

Table 4. Measured data for height of shoulder during walk by GA method

		Vertical variation	Average height (SD)
Front	Left	11.92	15.14(3.11)
	Right	12.66	15.08(3.19)
Back	Left	8.55	14.18(1.98)
	Right	7.49	14.13(1.99)
Total average		10.15	14.63(2.57)

Table 5. Measured data for height of shoulder during walk by GP method

		Vertical variation	Average height (SD)
Front	Left	10.03	15.23(2.44)
	Right	11.26	15.21(2.77)
Back	Left	7.71	14.03(1.72)
	Right	7.86	14.00(1.86)
Total average		9.22	14.62(2.20)

Table 4 includes variation and height of motion data for the GA method. The movements of the front legs are larger than of the rear legs in vertical variation. That is also verified through the larger standard deviations of average height. The average height of the front legs is higher than of the rear legs. The total average vertical variation is 10.15 cm and total average height is 14.63 cm.

Table 5 indicates same data for the GP method. The movements of the front legs and rear legs are similar to those of the GA method. The total average vertical variation is 9.22 cm and total average height is 14.62 cm. We see that the vertical variation is less than with the GA method, and height is almost the same.

4.3.3 Simulation Images of Gaits Obtained

Images of the robot walking are shown in Figure 4 in the Webots simulation. The obtained gaits of the best individuals for each method are displayed with frame images captured at intervals of 320ms. The images of the GA method show that the robot spreads its front legs out in turn, maintaining a low position. This gait shows relatively larger horizontal sway motions than the others.

The results of GP are somewhat similar to the GA results, but they show more balanced horizontal motion and the rear legs are used as well as the front legs.

Fig. 4. Simulation movement comparison between GA, GP (Best)

4.4 Bioloid Real Robot Results

Experiments for the real robot are executed through transferring simulation data of the best individual for each method into the controller of the Bioloid quadruped robot. Figure 5 shows experimental results for the two methods when executed on a hard stone floor. The images of the real walking of the quadruped robot are obtained from video clips at the same intervals as the simulation images.

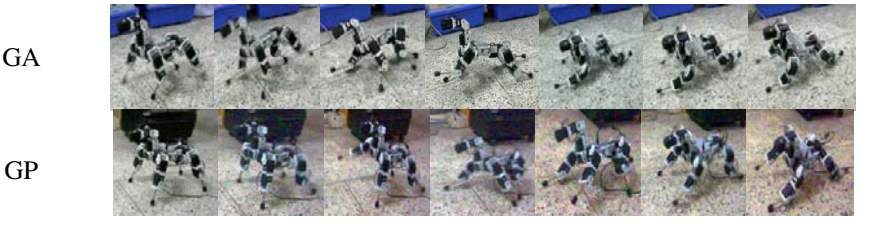


Fig. 5. Real movement comparison between GA, GP from Bioloid robots on the floor

The real walking with the GA method shows similar results to those from the simulation, relying primarily on the front legs for locomotion. However, there are some motions that result in some turning toward the right during walking. In comparison between frames 3 and 4 of the simulation in Figure 4, and same sequence in the real run in Figure 5, we can see the robot maintain its body without slipping in simulation, but not fully recovering its previous orientation after slippage in the real walking. This is what appears to be responsible for a slight right-turning tendency of the actual robot on the stone floor.

The results of the GP method appear more similar to the results in the simulation images than they did with the GA method. Most corresponding frames in the real and simulation runs appear almost identical. The only notable difference exists in the 3rd frame, in which the real robot raises its right front leg a little higher than it does in the simulation. Besides that, there is also some slippage during walking, but less than in the case of GA.

4.5 Analysis of Two Methods for Gait Evolution

The two methods (GA, GP) all seem to have some unique features for gait generation. The GA method provides intuitive understanding of gait shapes, because they are affected mainly by the locus of the paw. Therefore we can figure out some characteristics of gait based on analyzing the loci of paw positions. However, this approach depends heavily on the pre-defined shape of the paw locus, so is very different from global optimization.

The GP method with joint trajectory optimization has more possibility to reach global optimization because it is not dependent on a locus shape, but depends only on performance. However, it is difficult to obtain a global optimum because of the enormous size of the search space.

5 Conclusions

Two gait generation methods using GA (Genetic Algorithm), GP (genetic programming) are implemented for comparison to develop a fast locomotion scheme for a quadruped robot. Optimizations using the two proposed methods are executed and analyzed using the Webots simulation for a quadruped robot built by Bioloid.

In simulation, the GP method shows superior results in max velocity, while the GA methods show good performance in average speed.

In real experiments, the GA and GP approaches showed relatively similar movements to their respective simulation data. Especially, the GP method showed very similar movement between simulation and real experiment.

Further study will aim at refinement of the two methods to better reflect the characteristics of the gait problem and evolution of gait to minimize effects due to friction and slippage.

Acknowledgments

This work was supported by National Research Foundation of Korea Grant funded by the Korea government (2009-0071419).

References

1. Bekey, G.A.: Autonomous Robots-From Biological Inspiration to Implementation and Control. MIT Press, Cambridge (2005)
2. Busch, J., Ziegler, J., Aue, C., Ross, A., Sawitzki, D., Banzhaf, W.: Automatic generation of control programs for walking robots using genetic programming. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278, pp. 258–267. Springer, Heidelberg (2002)
3. Chernova, S., Veloso, M.: An Evolutionary Approach To Gait Learning For Four-Legged Robots. In: Proceedings of IROS 2004, Sendai, Japan, September 2004, pp. 2562–2567 (2004)
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, Beijing, China, October 9-15, pp. 5756–5761 (2006)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
6. Golubovic, D., Hu, H.: Parameter Optimisation of an Evolutionary Algorithm for On-line Gait Generation of Quadruped Robots. In: Proceedings of IEEE International Conference on Industrial Technology – ICIT 2003, Maribor, Slovenia, December 2003, pp. 221–226 (2003)
7. Hornby, G.S., Takamura, S., Yamamoto, T., Fujita, M.: Autonomous Evolution of Dynamic Gaits with Two Quadruped Robots. IEEE Trans. Robotics 21(3), 402–410 (2005)
8. Hohl, L., Tellez, R., Michel, O., Ijspeert, A.J.: Aibo and Webots: Simulation, wireless remote control and controller transfer. Robotics and Autonomous Systems 54(6), 472–485 (2006)
9. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
10. Mericli, T., Akın, H.L., Mericli, C., Kaplan, K., Celik, B.: The Cerbus 2005 Team Report
11. Robotis Corporation, `http://www.robotis.com`
12. Röfer, T.: Evolutionary Gait-Optimization Using a Fitness Function Based on Proprioception. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 310–322. Springer, Heidelberg (2005)
13. Seo, K., Hyun, S.: Genetic Programming Based Automatic Gait Generation for Quadruped Robots. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2008, Atlanta, July 2008, pp. 293–294 (2008)
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. Wang, Z.D., Wong, J., Tam, T., Leung, B., Kim, M.S., Brooks, J., Chang, A., Huben, N.V.: The 2002 rUNSWift Team Report (2002)

Markerless Localization for Blind Users Using Computer Vision and Particle Swarm Optimization

Hashem Tamimi and Anas Sharabati

Information Technology Department,
College of Administrative Sciences and Informatics,
Palestine Polytechnic University,
Ein Sara Str., P.O. Box 198, Hebron, Palestine
htamimi@ppu.edu

Abstract. In this paper, we propose a novel approach, which aims to solve the localization and target-finding problem for blind and partially sighted people. A guidance system, solely implemented on a mobile phone with a camera, is employed. A computer vision approach integrated with Particle Swarm Optimization (PSO) is proposed for tracking the location. Using PSO leads to many advantages: first, it is possible to obtain robust localization results by combining the current and historical information about the location of the blind person. Second, it helps the system to resolve from ambiguous situations caused by facing similar images at different locations. Third, it can detect and recover from cases where the calculated location is wrong. Experimental results show that the proposed method works efficiently because of the simplicity of the approach, which makes it suitable for mobile applications.

1 Introduction

People depend on sight to achieve most of their work, including determining their location. Blind people usually rely on other sensors, odometry, and artificial devices in order to achieve similar tasks. The walking cane is the most basic, inexpensive, and practical tool that accompanies the blind during his movement. Nowadays, blind persons face a more complicated world than before and the amount of help from sighted people is becoming less. Therefore, the use of technology to help the blind person is now of great need [2].

There have been many attempts and alternatives to guide the blind person. For example, in [18] a 3D-map of the environment is built and stored in a wearable computer, when the user walks in the environment, the system refers to the 3D-map and gives the user some directions, a body mounted camera is used for image acquisition. Another approach, proposed in [9], uses stereo images which enable depth detection; here also there is a need for a wearable computer and stereo vision for detecting depth.

Other ideas, such as [4] and [8], are based on augmenting the whole environment with RFID tags to give the blind an independent environment for themselves, These systems focus on giving the user information about the objects and

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 361–370, 2010.

about the way to reach them, but the environment must be fully pre-engineered with RFID tags. In [5], the place where the user walks contains artificial visual markers that can be detected by a mobile phone in order to guide the user. The problem in this approach is to guide the blind user to find the markers themselves.

From the above literature, it is obvious that there is a high demand for a low cost, light weight, and efficient alternative. In our work, we present a system that aims to aid the blind user in order to find her/his location by using a mobile phone with a camera (which is now available in most mobile phone models in the market). The system is supposed to communicate with the blind person through auditory massages. In addition, we aim to solve the localization problem indoors as a substitute for the GPS systems which are not reliable in closed places [3].

In the presented approach, a set of images are first captured from the environment under study, and the corresponding set of features are extracted from these images using a computer vision approach. These features, which implicitly represent the map of the environment, are stored in the memory of the mobile phone. Later, images from the location of the blind person are used as a query to determine the location. This is done by comparing their corresponding features with the implicit map. In order for the system to track the location of the user as she/he moves around in the environment, Particle Swarm Optimization is applied. In contrast with [5], the proposed system does not require any special visual artificial markers to be placed in the surroundings; it simply relies on the natural belongings of the environment. In addition to that, the proposed system can take into account the historical information to decide the location of the blind person. This makes the system more robust than the system proposed by [5] because the latter relies on a single image only.

This paper is organized as follows: section 2 presents the approach we have taken to solve the two main problems associated with constructing a navigational system. Section 3 introduces the features that we use in the system and the criteria of comparing them. Section 4 discusses the main algorithm that is used in the tracking of the location which is the PSO and shows the main advantages of using PSO for localization. Section 5 discusses the experiments and results of this paper.

2 The Establishment of the System

There are two phases for the proposed system. Phase 1 involves setting up the map and storing it in the mobile device. While Phase 2 deals with the steps of determining the location of the blind person as he moves. The two phases are explained below.

2.1 Phase 1

In this phase, images from the environment are captured using the mobile phone. For each image, or set of images, a given label that identifies the current location

is attached, and a short-voice massage is stored for each label. Then, a set of image based features are extracted from the images and stored in the mobile memory. These features are invariant to the expected image transformations that could occur during the motion of the blind person. For example, if we find a feature of some image, we expect that it will be similar to the feature of the rotated version of the same image to a high extent.

2.2 Phase 2

In this phase, the determination of the location for the blind person is carried out. First, new images are taken from a location. Then, features are extracted from these images. A comparison between the novel features and the stored features from Phase 1 is carried out. The label of most similar features in the database is identified and the corresponding voice message notifies to the user.

3 Color Histogram

There are many image-based features in the literature, such as Scale Invariant Feature Transform (SIFT) [11], integral invariants [17], orientation histogram [6]Ĕ etc. Nevertheless, we have chosen the color histogram in this work since it can be efficiently computed using the processor of the mobile phone. Color histograms are widely used as features for comparing images [12] because they can reflect the original image to a high extent and they are invariant to rotation and known to be very robust to occlusion. Although color histograms are sensitive to illumination changes found in outdoor applications, we avoid this problem by applying localization process indoors under limited illumination changes.

For an image $\mathbf{M}$ of size $N_0 \times N_1$, the histogram can be defined at a given intensity value $c = \mathbf{M}(x_0, x_1)$ as:

$$H(c) = \frac{1}{N_0 N_1} \sum_{x_0=0}^{N_0-1} \sum_{x_1=0}^{N_1-1} \delta\left(\mathbf{M}\left(x_0, x_1\right) - c\right). \tag{1}$$

In Equation 1, δ is the unitary impulse function. We notice that $H(c)$ values are normalized in order to sum to one. Elements of the histogram can be referred to as bins.

For comparing the histogram of the novel image with the previously stored histograms of Phase 1, we use Bhattacharyya distance[1]:

$$B(H_1, H_2) = \sqrt{1 - \sum_{1}^{\mathrm{N}} \sqrt{\sum \mathrm{H_{1i}} . \sum \mathrm{H_{2i}}}}. \tag{2}$$

Where H_1 and H_2 are two histograms, and N is the number of bins. The Bhattacharyya distance value is bounded between 0 and 1, where the value 0 refers for identical images. The larger the value the more the images are different.

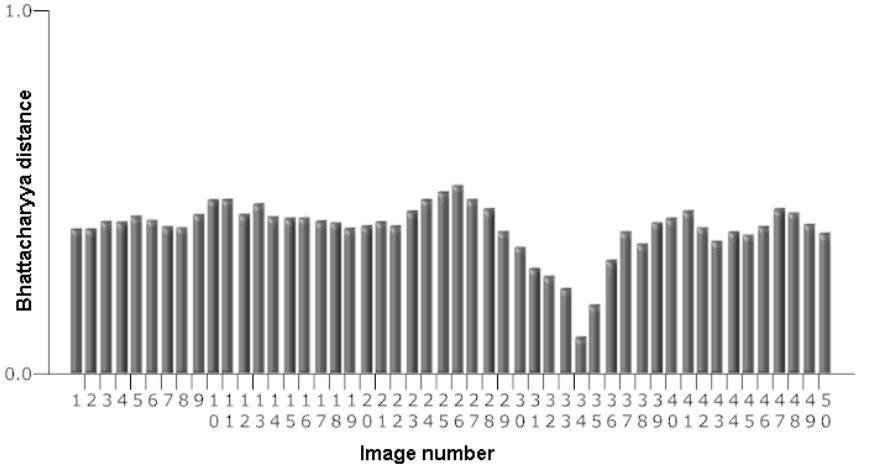


Fig. 1. Bhattacharyya distance chart, where X-axis represents the image number and the Y-axis represents the Bhattacharyya distance value

Figure 1 shows the Bhattacharyya distance function between a given histogram and another 50 histograms. The 50 histograms are taken from 50 different but overlapping images; this is why there is smoothness in the figure. We notice that the 34th element is a local minimum and therefore the corresponding image is the most similar to the image in the query.

In the localization process we can not rely on one image only because it will make the system fail when there are different locations with the similar visual appearances [14]. Therefore we must take into account both the current image as well as the historical data that leads to the current location. Such approaches are inherited from the robot localization process [15] where a decision about the location is calculated from the robot sensor at the current state as well as the previous decisions, Examples of such approaches are Particle Filter [16], and Particle Swarm Optimization [10]. We have adopted latter approach for its simplicity and effectiveness.

4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was proposed in 1995 by James Kennedy and Russell C. Eberhart [7,13]. The algorithm is inspired by the biological behavior of bird flocks, bees or fish during their search for food. In this algorithm, a number of particles search the space for a given target. Each particle searches its surrounding area for a local best solution and all the particles are affected by the best solution found, which is called the global best. The movement of the particles toward the goal is iterative and is controlled by the following equations:

$$\begin{aligned} v_i(t) = \omega \cdot v_i(t-1) + c_1 \cdot \alpha_1 \cdot (l_i - x_i(t-1)) \\ + c_2 \cdot \alpha_2 \cdot (g - x_i(t-1)), \end{aligned} \tag{3}$$

$$x_i(t) = x_i(t-1) + v_i(t). \tag{4}$$

Where i is the particle number, t is the iteration number, x_i is the position of the particle, v_i is the velocity of the particle, w, c_1, c_2 are scalar values, g is the global best value, l_i is a local best value of each particle, and α_1, α_2 are random values uniformly distributed between 0 and 1.

Equation 3 calculates the velocity of the particle, which indicates how many unites the particles move, whereas Equation 4 updates the position based on the velocity. At initialization all variables of the PSO are uniformly distributed in the search space.

The search space in the presented problem is one dimensional. One may assume that the searching process can be performed easily even sequentially with limited computation time. But the main reason of applying PSO here is the ability of the particles to establish a representation about the confidence of the correct location. Also, the particles can easily represent a motion model of the blind. The following subsections demonstrates some advantages of using PSO for localization:

4.1 Robust Localization

Initially, it is assumed that the particles are uniformly distributed all over the locations; this reflects a lost case since the probability for the blind to be at any location is equal. After applying Equation 3 and Equation 4 for a given number of iterations, the particles will move toward local minima, and the location with the maximum number of particles is assumed to be the right location of the blind person. When the blind person start moving, she or he is expected to be in a place not far from the previous position and the majority particles will start moving toward a new nearby local minima since the motion of particle is govern by the two equations above. This will eventually make the whole swarm follow the motion of the human robustly. In most cases, there will be no need to fully reinitializing the searching process each time a new location is encountered.

4.2 Ambiguous Locations

In real situations, it is possible to face similar images at different locations such as similar doors, windows or similar furniture at offices or hospitals. In such cases, the localization process depending on one-shot-decision can lead to wrong location. Another reason for this to happen is the weakness of the feature extraction algorithm to find unique features for dissimilar images. In both cases, it is not recommended to rely on the current location in isolation of the previous location(s) that leads to it during the motion of the blind person. One major advantage of applying PSO for localization is that the current decision about the location of the blind is actually an accumulation from the historical information. PSO memorizes and preserves the context that leads to a current behavior, which makes it very suitable for this application.

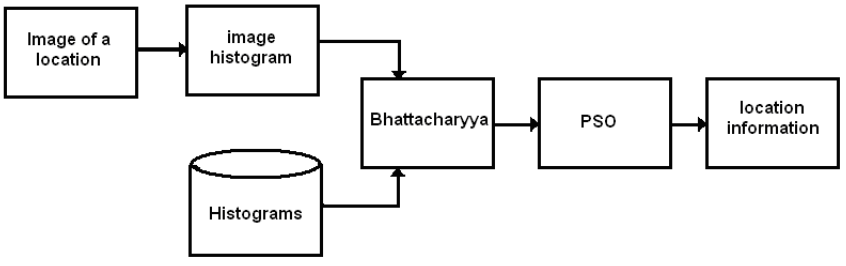


Fig. 2. The system steps for each image received

4.3 Lost Cases

In addition to preserving the context of the motion of the blind person, another advantage of the proposed system is the ability to detect and recover from the cases where it gets lost. This is important in order not to miss-guide the blind person. We define a lost case is a non-initial case where the Bhattacharyya distance of each particle larger than a given threshold. If this happens it means that the particles are giving the wrong directions or that the environment is not pre-defined. To recover from this case, the PSO algorithm will fully restart the search process by re-distributing the particles to whole search space and then tries to find the correct location again.

Figure 2 shows the block diagram of the proposed system. It is worth mentioning that communication with the blind user does not occur immediately after a single query image but after capturing a given number of images in order for the particles to gather at the right location. The number of images that is required for this purpose is found experimentally as explained in the experimental part of this paper.

5 Experiments and Results

The experiments were done with a simulation program which accepted training histograms as explained in Phase 1 and test images as explained in Phase 2. The samples were collected form a mid-size grocery store with 16 different locations. We used a set of 2748 images in the experiment. Several types of color histograms were studied, namely grayscale histograms with 64, 128, and 256 bins as well as a 3D-histogram for colored images with 8 bins for each color intensity. The experiment involved a simulation of a moving person and the images that faced her/him during her/his motion were identified. This is clarifed in Figure 3, where the *Start* and the *End* positions of an image sequence (path) were identified. The number of particles, the PSO constants, and the PSO iterations were also adjustable in the program. Once these parameters are entered, the simulation showed us the images that corresponds to blind person location as she/he moves

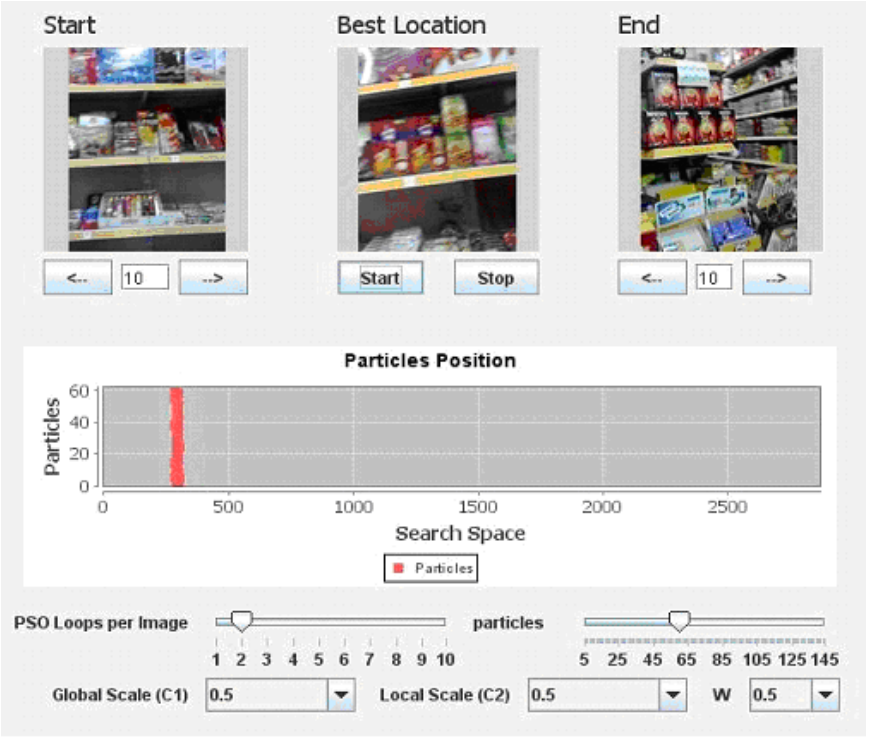


Fig. 3. The simulation program

from the staring position to the ending position. This is referred to as the *Best Location* in Figure 3. Based on PSO, this *Best Location* is the one which has the maximum number of particles.

Several experiments with different paths were carried out. For each path, the number of images (N) needed to find a given location, and the number of times the system failed to find the right location (E) were recorded, (this happened when the system needs to re-initialize the particles). In addition, the cases were the system completely failed to find a path (-) is recorded. Table 1(a) to Table 1(b) show the results of these experiments using different histograms.

From Table 1(a) to Table 1(c) we can see that grayscale histograms failed most of the time to find and follow the test path, and to correct itself in case of a failure in localization. In large environments with similar location the grayscale histogram cannot distinguish the images. On the other hand, a 3D histogram in Table 1(d) with 8 bins for each intensity succeeded to follow all the test paths, which means it is the best histogram to use.

In order to test the system ability to recover from faults, some faults were intentionally entered in the test paths by inserting a number of images that did not belong to a given image sequence. The system was able to detect the these faults and recover from them.

Table 1. Experimental Results using different histograms

(a) Grayscale 64 bin histogram

Test path	N	E	Reach Destination?
Location 1 to 2	1	2	No
Location 2 to 4	3	0	Yes
Location 6 to 4	4	0	Yes
Location 8 to 10	2	0	Yes
Location 12 to 10	-	-	No
Location 15 to 12	-	-	No
Location 14 to 12	-	-	No
Location 14 to 16	-	-	No
Success Rate			37.5%

(b) Grayscale with 128 bin histogram

Test path	N	E	Reach Destination?
Location 1 to 2	1	1	No
Location 2 to 4	2	0	Yes
Location 6 to 4	3	0	Yes
Location 8 to 10	3	0	Yes
Location 12 to 10	4	0	Yes
Location 15 to 12	3	0	Yes
Location 14 to 12	-	-	No
Location 14 to 16	-	-	No
Success Rate			62.5%

(c) Grayscale with 256 bin histogram

Test path	N	E	Reach Destination?
Location 1 to 2	1	1	No
Location 2 to 4	2	0	Yes
Location 6 to 4	4	0	Yes
Location 8 to 10	4	0	Yes
Location 12 to 10	-	-	No
Location 15 to 12	4	0	Yes
Location 14 to 12	3	0	Yes
Location 14 to 16	-	-	No
Success Rate			62.5%

(d) Color 3D histogram with 8 bins

Test path	N	E	Reach Destination?
Location 1 to 2	1	0	Yes
Location 2 to 4	2	0	Yes
Location 6 to 4	3	0	Yes
Location 8 to 10	3	0	Yes
Location 12 to 10	3	0	Yes
Location 15 to 12	3	0	Yes
Location 14 to 12	3	0	Yes
Location 14 to 16	3	0	Yes
Success Rate			100.0%

From Table 1(d), we can also see that the number of images needed to follow the test path correctly is 3. This means we can apply the system efficiently because most cameras in the mobile phones can run in a rate of 10 to 30 frames/sec.

In addition to the above experiments, we made some other experiments that involved testing the system under illumination changes. For example, we have tested the proposed system in places near windows or doorway allowing the sunlight to affect the images. It was obvious that the system did not find the right locations in most cases that involved illumination changes. In order to overcome this problem, we have observed that the system reports these locations as being in a lost location. Therefore, the blind person has the ability to move the camera nearer to the location, if she or he was notified by the system.

6 Conclusion

In this paper we proposed a new approach for guiding a blind person in indoor environment. The proposed system has many advantages such as: it can be developed on a mobile phone that has an embedded camera which means that the user does not need to carry heavy equipments. It relies on historical information to find the location which is better than depending on one image, because depending on history solves the problem of similar places in the environment. It has low setup cost and does not involve the placement of tags or any other markers in the environment.

In the experimental work, we have concluded that the integration of PSO with 3D color histogram is more successful than the gray scale histograms. The proposed system is robust, it can detect and recover from the lost cases, and therefore does not miss-guide the blind user. The problem of illumination is to be further investigated in the future.

References

1. Aherne, F., Thacker, N., Rockett, P.I.: The bhattacharyya metric as an absolute similarity measure for frequency coded data. Kybernetika 4(4), 363–368 (1998)
2. Brabyn, J.: Technology as a support system for orientation and mobility. The Free Library, September 22 (1997)
3. Chiu, D., O'Keefe, K.: Seamless outdoor-to-indoor positioning. GPS World 20(3), 32–38 (2009)
4. Coroama, V., Röthenbacher, F.: The chatty environment - providing everyday independence to the visually impaired. In: Workshop on ubiquitous computing for pervasive healthcare applications at UbiComp 2003, Seattle (October 2003)
5. Coughlan, J., Manduchi, R.: Functional assessment of a camera phone-based wayfinding system operated by blind and visually impaired users. International Journal on Artificial Intelligence Tools 18(3), 379–397 (2009)
6. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 428–441. Springer, Heidelberg (2006)
7. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Service Center, Nagoya (1995)
8. Hub, A., Diepstraten, J., Ertl, T.: Augmented indoor modeling for navigation support for the blind. In: The International Conference on Computers for People with Special Needs (CPSN 2005), Las Vegas, pp. 54–62 (2005)
9. Hub, A., Hartter, T., Ertl, T.: Interactive tracking of movable objects for the blind on the basis of environment models and perception-oriented object recognition methods. In: Assets 2006: Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility, pp. 111–118. ACM, New York (2006)
10. Kronfeld, M., Weiß, C., Zell, A.: A dynamic swarm for visual location tracking. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 203–210. Springer, Heidelberg (2008)
11. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal on Computer Vision 60(2), 91–110 (2004)
12. Pass, G., Zabih, R.: Histogram refinement for content-based image retrieval. In: Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision (WACV), pp. 96–102. IEEE Computer Society Press, Washington (1996)
13. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence 1(1), 33–57 (2007)
14. Sonnoqrot, F., Younis, E., Tamimi, H.: Vision-based localization aid for the blind. In: Palestinian International Conference on Computer and Information Technology (PICCIT), Hebron, Palestine, September 1-3 (2007)
15. Tamimi, H.: Vision-based features for mobile robot localization. Ph.D. thesis, University of Tüebingen. Tuebingen, Germany (2006)

16. Tamimi, H., Andreasson, H., Treptow, A., Duckett, T., Zell, A.: Localization of mobile robots with omnidirectional vision using particle filter and iterative sift. In: Proceedings of the European Conference on Mobile Robots (ECMR), Ancona, Italy, pp. 1–7 (2005)
17. Tamimi, H., Halawani, A., Burkhardt, H., Zell, A.: Appearance-based localization of mobile robots using local integral invariants. In: Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-9), Tokyo, Japan (March 2006)
18. Treuillet, S., Royer, E., Chateau, T., Dhome, M., Lavest, J.: Body mounted vision system for visually impaired outdoor and indoor wayfinding assistance. In: Proceedings of the Conference and Workshop on Assistive Technologies for People with Vision and Hearing Impairments: Assistive Technology for All Ages (CVHI 2007), Granada, Spain, August 28-31 (2007)

Particle Swarm Optimization for Feature Selection in Speaker Verification

Shahla Nemati and Mohammad Ehsan Basiri

Young Research Club, Islamic Azad University, Arsanjan Branch, fars, Iran
Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Hezar Jerib Ave., 81744 Isfahan, Iran
nemati@iaua.ac.ir
basiri@eng.ui.ac.ir

Abstract. The problem addressed in this paper concerns the feature subset selection for an automatic speaker verification system. An effective algorithm based on particle swarm optimization is proposed here for discovering the best feature combinations. After feature reduction phase, feature vectors are applied to a Gaussian mixture model which is a text-independent speaker verification model. The performance of proposed system is compared to the performance of a genetic algorithm-based system and the baseline algorithm. Experimentation is carried out, using TIMIT corpora. The results of experiments indicate that with the optimized feature subset, the performance of the system is improved. Moreover, the speed of verification is significantly increased since by use of PSO, number of features is reduced over 85% which consequently decrease the complexity of our ASV system.

Keywords: Particle Swarm optimization (PSO), Feature Selection (FS), Speaker Verification, Gaussian Mixture Model (GMM), Genetic Algorithm (GA).

1 Introduction

Automatic speaker verification (ASV) systems are designed to answer the question "Is the present speaker the one s/he claims to be, or not?". The verification process terminates with a binary decision (access is granted or not) that depends on the degree of similarity between the speech sample and a predefined model for the user, whose identity the speaker claims [1].

There are many applications to speaker verification; Access control to facilities, secured transactions over a network, electronic commerce applications, forensic application and telephone banking are the most important ones [2].

ASV refers to the task of verifying speaker's identity using speaker-specific information contained in speech signal. Low-level acoustic features contain some

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 371–380, 2010.

redundant features, which can be eliminated using feature selection (FS) techniques. The purpose of FS is to reduce the maximum number of irrelevant features while maintaining acceptable classification accuracy. By doing that, it also reduces redundancy in the information provided by the selected features [3].

Feature selection is of considerable importance in signal processing [4], bioinformatics [5], text categorization [6], data mining and pattern recognition [3]. FS has been occasionally used in ASV systems. Nemati et al. [4] used an ant colony optimization (ACO) algorithm to select relevant features for text-independent speaker verification; Day and Nandi [7] employed genetic programming (GP) for FS; also Pandit and Kittkr [8] used L plus-R minus feature selection algorithm for text-dependent speaker verification. Furthermore, Cohen and Zigel [9] employed dynamic programming for FS in speaker verification and Ganchev et al. [1] used information gain (IG) and gain ratio (GR) for FS in ASV task.

Among too many methods that are proposed for FS, population-based optimization algorithms such as genetic algorithm (GA), ACO and particle swarm optimization (PSO) have attracted a lot of attention [5,10,11]. These methods are stochastic optimization techniques attempt to achieve better solutions by application of knowledge from previous iterations.

Particle swarm optimization is a population-based stochastic optimization technique inspired by behavior of natural swarms and was developed by Kennedy and Eberhart in 1995 [12,13]. PSO has many advantages over other evolutionary computation techniques (for example, genetic algorithms) such as simpler implementation, faster convergence rate and fewer parameters to adjust [14].

In this paper, we propose a PSO-based algorithm for feature selection and apply it to feature vectors containing mel-frequency cepstral coefficients (MFCCs) and their delta coefficients, two energies, linear prediction cepstral coefficients (LPCCs) and their delta coefficients. Then, feature vectors are applied to a Gaussian mixture models (GMM), which is a text-independent speaker verification model. Finally, verification performance and the length of selected feature vectors of proposed PSO-based algorithm are compared to a GA-based and the baseline algorithm for performance evaluation. To the best of our knowledge, it is the first paper to apply a particle swarm optimization technique to the problem of feature selection in ASV systems.

The rest of this paper is organized as follows. Section 2 presents a brief overview of ASV systems. Proposed system is described in Section 3. Section 4 reports computational experiments. It also includes a brief discussion of the results obtained and finally the conclusion and future works are offered in the last section.

2 An Overview of ASV Systems

The typical process in most proposed ASV systems involves some form of preprocessing of the data (silence removal) and feature extraction, followed by some form of speaker modeling to estimate class dependent feature distributions. These steps are described in following sections in more details.

2.1 Feature Extraction

The original signal, the speech waveform, comprises all information about the speaker, and each step in the extraction process can only decrease the mutual information or leave it unchanged. The objective of feature extraction is to reduce the dimension of the extracted vectors and thereby reduce the complexity of the system. The main task for the feature extraction process is to pack as much speaker-discriminating information as possible into as few features as possible.

The choice of features in any proposed ASV system is of primary concern, because if the feature set does not yield sufficient information then trying to estimate class dependent feature distributions is pointless [10]. Most commonly used feature extraction techniques, such as MFCCs and LPCCs have been particularly popular for ASV systems in recent years. This transforms give a highly compact representation of the spectral envelope of a sound [15].

2.2 Feature Selection

Feature selection is a problem of global combinatorial optimization in machine learning, which reduces the number of features, removes irrelevant, noisy and redundant data, and results in acceptable classification accuracy.

Searching for an optimal feature subset is known to be an NP-complete problem. Usually FS algorithms involve heuristic or random search strategies in an attempt to avoid this prohibitive complexity. However, the degree of optimality of the final feature subset is often reduced [3].

The objectives of feature selection are manifold, the most important ones being: improving models performance, providing faster and more cost-effective models, and to obtain a profounder insight into the underlying processes that generated the data. Some surveys of feature selection algorithms are given in [16,17].

2.3 Speaker Modeling

The speaker modeling stage of the process varies more in the literature. The aim of speaker modeling is to distinguish an individual that is enrolled into an ASV system with the aim of defining a model (usually feature distribution values). The most popular methods in the literature are GMM [15], and vector quantization (VQ). Other techniques such as decision trees, support vector machine (SVM) and artificial neural network (ANN) have also applied [4]. In this paper, GMM is used for speaker modeling.

3 Proposed System

Fig. 1 shows the overall process of the proposed system. To apply a PSO algorithm to solve a feature selection problem, some aspects need first to be addressed.

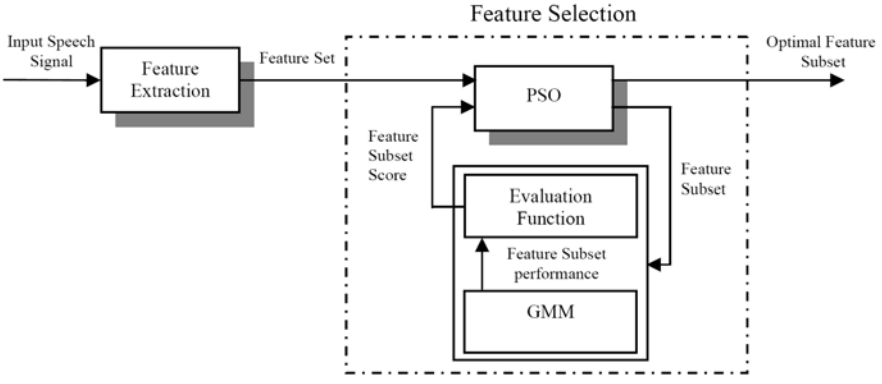


Fig. 1. Overall process of PSO feature selection for ASV

3.1 Representation of Position

PSO is initialized with N particles moving around in a D-dimensional search space. The position of the ith particle at tth iteration is represented by $X_i^t = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The best position encountered by a particle (*pbest*) denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ and the best position encountered by the whole swarm (*gbest*) denoted as $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$. Every bit represents a feature and the value '1' indicates the selection of corresponding feature. Therefore, each position is a feature subset [12].

3.2 Representation of Velocity

Every particle's velocity is represented as a positive integer, varying between 1 and *Vmax*. It implies how many of the particle's bits (features) should be changed, to be the same as that of the global best position. The number of different bits between two particles relates to the difference between their positions. The rate of the position change (velocity) for particle i at iteration t is represented as $V_i^t = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

3.3 Position Update Strategies

The particle successively adjusts its position toward the global optimum according to *pbest* and *gbest*. The particles are manipulated according to the Equations (2) and (3)

$$v_{id}(t+1) = w \times v_{id}(t) + c_1.r_{1d}(t).[p_{id}(t) - x_{id}(t)] + c_2.r_{2d}(t).[p_{gd}(t) - x_{id}(t)], \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), \quad (2)$$

where w is the inertia weight; it is a positive linear function of time changing according to the generation iteration. The acceleration constants c_1 and c_2 in Equation (2) represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. r_{1d} and r_{2d} are two random functions in the range [0, 1].

After updating the velocity, a particle's position will be updated by the new velocity. Assume that the new velocity is V; the number of different bits between the current particle and *gbest* is n_g. If $V \leq n_g$, V bits of the particle are randomly changed, different from that of *gbest*. The particle then moves toward the global best while still exploring the search space, instead of simply being same as *gbest*. If $V > n_g$, In addition to changing all the different bits to be same as that of *gbest*, we should further randomly change $(V - n_g)$ bits outside the different bits between the particle and *gbest*. So after the particle reaches the global best position, it keeps on moving some distance toward other directions, enabling further search.

3.4 Fitness Function

In our experiments, the fitness function is defined according to Equation (4)

$$Fitness = \alpha \times \psi_S + \beta \times \frac{|N| - |S|}{|N|}, \tag{3}$$

where $\psi(S)$ is classifier performance for the feature subset S, $|N|$ is the total number of features, $|S|$ is feature subset length, $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$. This formula means that the classifier performance and feature subset length have different significance for feature selection task. In our experiment we assume that classification quality is more important than subset length and we choose $\alpha = 0.8$ and $\beta = 0.2$.

3.5 Proposed Feature Selection Algorithm

The main steps of proposed feature selection algorithm are shown in Fig. 2. The process begins by generating a population of particles with random positions in the search space. In the next step Equation (4) is used to evaluate each particle. *pbest* and *gbest* are calculated at the end of each iteration and if the quality of solution found by a particle is higher than its previous *pbest*, this solution will be the new *pbest* for that particle and the best *pbest* among all particles is selected as *gbest*. In the next step, stop criteria are tested; if an optimal subset has been found or the algorithm has executed a certain number of times, then the process halts and outputs the best feature subset encountered. If none of these conditions hold, then position and velocity of particles are updated according to Equations (2) and (3) then, the process iterates once more.

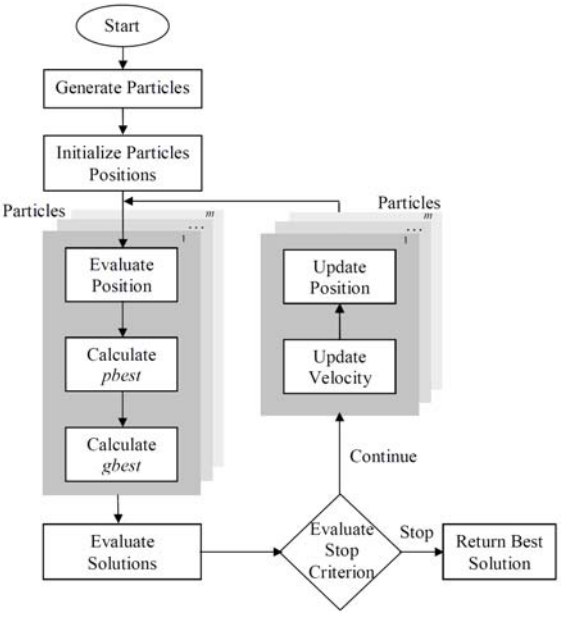


Fig. 2. PSO-based feature selection algorithm

4 Experimental Results

4.1 TIMIT Dataset

The TIMIT corpus [18] is used in this paper. This corpus contains 630 speakers (438 male and 192 female) representing 8 major dialect regions of the United States, each speaking ten sentences. There are two sentences that are spoken by all speakers and the remaining eight are selected randomly from a large database. The 100 speakers included in both the test and train directories were used during the TIMIT(100) trials.

4.2 Evaluation Measure

We evaluated our ASV system with detection error tradeoff (DET) curves, which show the tradeoff between false alarm (FA) and false rejection (FR) errors. Typically equal error rate (EER), which is the point on the curve where FA = FR, is chosen as evaluation measure. We also used detection cost function (DCF) defined as

$$DCF = C_{miss}.FRR.P_{target} + C_{FA}.FAR.(1 - P_{target}), \tag{4}$$

where P_{target} is the priori probability of target tests with $P_{target} = 0.1$, FRR and FAR are false rejection rate and false acceptance rate respectively at an operating point and the specific cost factors $C_{miss} = 10$ and $C_{FA} = 1$ [4].

Table 1. GA and PSO parameter settings

	Population	*Iteration*	*Crossover Prob.*	*Mutation Prob.*	c_1	c_2	w	V_{max}
GA	30	50	0.7	0.3	-	-	-	-
PSO	30	50	-	-	1	1	0.4-1.4	1-17

4.3 Experimental Setup

Various values were tested for the parameters of GA and proposed algorithm. The experiments show that the highest performance is achieved by setting the control parameters to values shown in Table 1. Experiments were conducted on a subset of TIMIT corpora consist of 60 male and 40 female speakers of different accent that were selected randomly. Data were processed in 20 ms frames (320 samples) with 50% overlaps. Frames were segmented by Hamming window and pre-emphasized with $\alpha = 0.97$ to compensate the effect of microphone's low pass filter. At first, feature vector was created by extracting MFCCs from silence-removed data for each frame. In the next step, delta coefficients were calculated based on the MFCCs and appended to existing feature vector. Furthermore, two energies were applied to input vectors as described earlier. Then we consider the LPCCs and their delta coefficients respectively and append them to the feature vector. The final feature set contains $F = 50$ features. Finally, verification process was performed using the GMM approach with different number of Gaussian (16, 32). The performance criterion is due to EER and DCF according to an adopted decision threshold strategy.

4.4 Results

The verification quality and feature subset length are two criteria that are considered to assess the performance of algorithms. Comparing the first criterion, we noted that both PSO-based and GA-based algorithms reduce the dimensionality of feature space. Furthermore, the PSO-based algorithm selects a smaller subset of features than the GA-based algorithm. Table 2 shows the number of selected features by PSO-based and GA-based approaches. As we can see in Table 2, PSO can degrade dimensionality of features over 85%. The second performance criterion is due to EER and DCF according to an adopted decision threshold strategy. The EER and DCF for baseline algorithm (using all possible features)

Table 2. Selected Features of PSO-GMM

Selection Method	Number of selected features	Percentages of selected features
PSO-GMM(16)	9	18%
PSO-GMM(32)	7	14%
GA-16	16	32%
GA-32	17	34%

Table 3. EER and DCF for three algorithms with different number of Gaussians

Number of	GMM		PSO		GA	
Gaussians	EER	DCF	EER	DCF	EER	DCF
16	5.18	0.0571	4.907	0.0491	4.866	0.0500
32	4.965	0.0501	3.634	0.0312	4.56	0.0467

PSO-based and GA-based algorithms with different number of Gaussian (16, 32) were shown in Table 3.

DET curves for GA-based and PSO-based algorithms with 16, 32 Gaussians are shown in Fig. 3. From the results, it can be seen that PSO-GMM yields a significant improvement in speed than the baseline GMM approach. The improvement is due to the selection of optimal feature set by PSO algorithm.

4.5 Discussion

Experimental results show that the use of unnecessary features decrease verification performance and hurt classification accuracy and FS is used to reduce redundancy in the information provided by the selected features. Using only a small subset of selected features, the PSO and the GA algorithms obtained better verification accuracy than the baseline algorithm using all features.

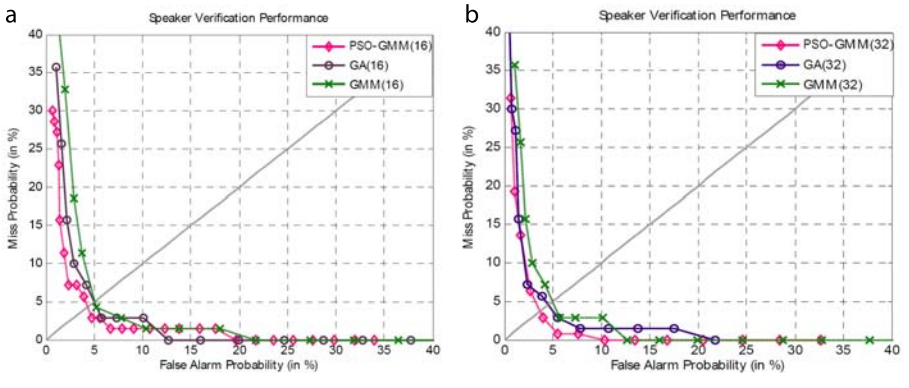


Fig. 3. DET curves for GMM, PSO-GMM and GA with (a) 16 Gaussians and (b) 32 Gaussians

Both PSO and GA are stochastic population-based search approaches that depend on information sharing among their population members to enhance their search processes using a combination of deterministic and probabilistic

rules. They are efficient, adaptive and robust search processes, producing near optimal solutions, and have a large degree of implicit parallelism. The main difference between PSO compared to GA, is that PSO does not have genetic operators such as crossover and mutation. Particles update themselves with the internal velocity; they also have a memory that is important to the algorithm.

In GAs, chromosomes share information with each other, so the whole population moves like one group towards an optimal area. In PSO, only the 'best' particle gives out the information to others. It is a one-way information sharing mechanism, the evolution only looks for the best solution. Compared with GAs, all the particles tend to converge to the best solution quickly even in the local version in most cases.

Compared to GAs, the PSO has a much more intelligent background and can be implemented more easily. The computation time used in PSO is less than in GAs. The parameters used in PSO are also fewer. However, if the proper parameter values are set, the results can easily be optimized. The decision on the parameters of the particle swarms affects the exploration-exploitation tradeoff and is highly dependent on the form of the objective function [19]. Successful feature selection was obtained even using conservative values for the PSO basic parameters [12].

5 Conclusion and Future Research

In this paper, we have addressed the problem of optimizing the acoustic feature set by PSO algorithm for text-independent speaker verification system based on GMM. Original feature set contains MFCCs and their delta coefficients, two energies, LPCCs and their delta coefficients. Proposed PSO-based feature selection algorithm selects the most relevant features among all features in order to increase the performance of our ASV system. We compare its performance with another prominent population-based feature selection method, genetic algorithm. The experimental results on a subset of TIMIT database showed that PSO selects the more informative features without losing the performance than GA. The feature vectors size reduced over 85% that led to a less complexity of our ASV system. Moreover, verification process in the test phase speeds up because less complexity is achieved by the proposed system in comparison with current ASV systems.

PSO has the ability to converge quickly; it has a strong search capability in the problem space and can efficiently find minimal feature subset. Experimental results demonstrate competitive performance. More experimentation and further investigation into this technique may be required.

For future work, the authors plan to investigate the performance of proposed ASV system by taking advantage of using VQ, GMM-UBM and other models instead of GMM. Another research direction will involve the consideration of utilizing hybrid approaches like PSO-SVM, PSO-GA and other hybrid algorithms for feature selection in ASV systems.

References

1. Ganchev, T., Zervas, P., Fakotakis, N., Kokkinakis, G.: Benchmarking Feature Selection Techniques on the Speaker Verification Task. In: Fifth International Symposium on Communication Systems, Networks And Digital Signal Processing, pp. 314–318 (2006)
2. Bimbot, F., et al.: A Tutorial on Text-Independent Speaker Verification. EURASIP Journal on Applied Signal Processing 4, 430–451 (2004)
3. Jensen, R.: Combining rough and fuzzy sets for feature selection. Ph.D. Thesis, University of Edinburgh (2005)
4. Nemati, S., Boostani, R., Jazi, M.D.: A Novel Text-Independent Speaker Verification System Using Ant Colony Optimization Algorithm. In: Elmoataz, A., Lezoray, O., Nouboud, F., Mammass, D. (eds.) ICISP 2008. LNCS, vol. 5099, pp. 421–429. Springer, Heidelberg (2008)
5. Nemati, S., Basiri, M.E., Ghasem-Aghaee, N., Aghdam, M.H.: A novel ACO-GA hybrid algorithm for feature selection in protein function prediction. Expert systems with applications 36, 12086–12094 (2009)
6. Aghdam, M.H., Ghasem-Aghaee, N., Basiri, M.E.: Text Feature Selection using Ant Colony Optimization. Expert systems with applications 36, 6843–6853 (2009)
7. Day, P., Nandi, A.K.: Robust Text-Independent Speaker Verification Using Genetic Programming. IEEE Transactions on Audio, Speech, and Language Processing 15, 285–295 (2007)
8. Pandit, M., Kittkr, J.: Feature Selection for a DTW-Based Speaker Verification System, pp. 796–799. IEEE, Los Alamitos (1998)
9. Cohen, Zigel, Y.: On Feature Selection for Speaker Verification, of COST 275 workshop on The Advent of Biometrics on the Internet (2002)
10. Basiri, M.E., Ghasem-Aghaee, N., Aghdam, M.H.: Using ant colony optimization-based selected features for predicting post-synaptic activity in proteins. In: Marchiori, E., Moore, J.H. (eds.) EvoBIO 2008. LNCS, vol. 4973, pp. 12–23. Springer, Heidelberg (2008)
11. Liu, Y., Qin, Z., Xu, Z., He, X.: Feature Selection with Particle Swarms. In: Zhang, J., He, J.-H., Fu, Y. (eds.) CIS 2004. LNCS, vol. 3314, pp. 425–430. Springer, Heidelberg (2004)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceeding of IEEE International Conference on Neural Networks, pp. 1942–1947. IEEE Press, Perth (1995)
13. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
14. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization an overview. J. Swarm Intelligence 1, 33–57 (2007)
15. Cheung-chi, L.: GMM-Based Speaker Recognition for Mobile Embedded Systems, Ph.D. Thesis, University of Hong Kong (2004)
16. Mladenič, D.: Feature Selection for Dimensionality Reduction. In: Saunders, C., Grobelnik, M., Gunn, S., Shawe-Taylor, J. (eds.) SLSFS 2005. LNCS, vol. 3940, pp. 84–102. Springer, Heidelberg (2006)
17. Bins, J.: Feature Selection from Huge Feature Sets in the Context of Computer Vision. Ph.D. dissertation, Department Computer Science, Colorado State University (2000)
18. Garofolo, J., et al.: DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM. National Institute of Standards and Technology (1990)
19. Trelea, I.C.: The particle swarm optimization algorithm: Convergence analysis and parameter selection. J. Inf. Process. Lett. 85, 317–325 (2003)

Scale- and Rotation-Robust Genetic Programming-Based Corner Detectors

Kisung Seo and Youngkyun Kim

Dept. of Electronic Engineering, Seokyeong University Seoul, Korea

Abstract. This paper introduces GP- (Genetic Programming-) based robust corner detectors for scaled and rotated images. Previous Harris, SUSAN and FAST corner detectors are highly efficient for well-defined corners, but frequently mis-detect as corners the corner–like edges which are often generated in rotated images. It is very difficult to avoid incorrectly detecting as corners many edges which have characteristics similar to corners. In this paper, we have focused on this challenging problem and proposed using Genetic Programming to do automated generation of corner detectors that work robustly on scaled and rotated images. Various terminal sets are presented and tested to capture the key properties of corners. Combining intensity-related information, several mask sizes, and amount of contiguity of neighboring pixels of similar intensity, allows a well-devised terminal set to be proposed. This method is then compared to three existing corner detectors on test images and shows superior results.

Keywords: corner detector, genetic programming, automated generation, scale and rotation robust.

1 Introduction

Corner detection is used for various applications such as tracking, SLAM (simultaneous localization and mapping), image matching and recognition.

Many algorithms for corner detection have been introduced in computer vision research. Moravec [8] developed an interest point detector based on the autocorrelation function. He measured the differences between the feature window and its shifted versions. Kitchen and Rosenfeld [6] proposed a corner measure based on the change of gradient direction along an edge contour at the local gradient magnitude. Harris and Stephens [5] modified the Moravec method and proposed the famous Harris corner detector by estimating the autocorrelation from the first-order derivatives.

Different types of corner detectors examine the area morphologically and try to locate corner-like shapes. Smith and Brady [13] proposed a corner detection algorithm known as the SUSAN corner detector based on brightness comparison and a corner mask in a circular window. Rosten and Drummond [10] introduced the FAST (Features from Accelerated Segment Test) feature detector. This is sufficiently fast that it allows on-line operation of a tracking system. To optimize the detector for generality

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 381–391, 2010.

and speed, this model is used to train a decision tree classifier. This recent approach shows good performance in repeatability and efficiency in real world application.

However, those competitive algorithms may detect false corners when the boundaries are step-like or there is a small change in slope which just represents an edge. Edge pixels have similar characteristic with corner pixels, so this phenomenon can occur after reduction and rotation of images. In the other hand, most corner detectors are hand-coded designs, product of the analysis and interpretations of how the problem has been confronted by a human mind.

On the other hand, the importance of automated synthesis approaches has been growing. GP [7] is an approach to automatic programming, in which a computer can construct and refine its own programs to solve specific tasks. Zhang [15, 16] uses GP for a number of object classification and detection problems. Good results have been achieved on classification and detection of regular objects against a relatively uncluttered background.

There exist some contributions directly related to our work, and specifically addressing the problem of corner detector. Ebner [3, 4] posed interest point detection as an optimization problem, and attempted to evolve the Moravec interest point operator [8] using Genetic Programming. The approach by Ebner fails to capture the essence of the desired properties that a general interest point detector should attempt to fulfill. The shortcomings of these contributions are overcome in Trujillo and Olague's work [9, 14] by realizing a thorough analysis in order to define a suitable measure of the stability, as given by the operators' repeatability rates; as well as by quantification of the operators' global separability. However, the results were compared only with the Harris detector, and showed just similar performance.

We propose Genetic Programming-based automated generation of corner detectors which is robust to scale and rotational transformation. Various terminal sets are introduced to capture the key properties of corners, combining intensity-related information, several mask sizes, and contiguity of neighboring pixels allows a well-devised terminal set to be proposed. The proposed method is compared to existing corner detectors on scaled and rotated test images.

2 Problem and Proposed Method

2.1 Problem Statements

Most existing algorithms perform well for well-defined corners in images (top panels in Figure 1). However, some corners are ill-defined because of noise and some non-corners are often misclassified as corners for example, inclined edges (bottom panels in Figure 1. Especially corner-like edges are often generated when an image is rotated. It is very difficult to detect correctly these edges which have characteristics similar to corners. In this paper, we have focused on the challenging problem above and proposed a Genetic Programming-based automated generation of corner detectors that is robust to scaling and rotation of images.

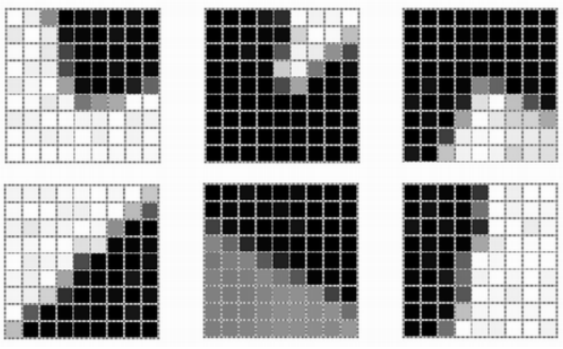


Fig. 1. Corners (top) and edges (bottom) on an image

2.2 Genetic Programming

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 corner detectors, as described below. To evolve corner detectors, we need to specify the set of terminals and the set of elementary functions.

Besetti and Soule [1] examine different functions sets for the linear regression problem to begin to understand how choice of function sets influences the search process. They found that different reasonable function sets can have a significant effect on performance. We believe that this phenomenon might vary strongly according to the domain or characteristics of the problems to be treated. In our experience, it is very important in corner detection to define function and terminal sets carefully. The terminal set seems to be especially critical, because it is more closely related to physical characteristics of corner points than is the function set, which usually represents general-purpose operators.

2.3 GP Function and Terminal Set

The function set for the proposed GP-based detector involves 5 arithmetic operator and 3 conditional operators as follows.

$$F = \{ +, -, \times, \div, \text{abs}, \text{if}, \min, \max \} \quad (1)$$

The set of primitive functions has to 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. For corner detection, the choice of function set seems to be less critical than the choice of terminal set.

Several efforts have been made to design an appropriate terminal set for corner detection. Ebner [3, 4] selected a gray scale representation of the input image as the sole terminal. The image intensities were scaled to the range [0,1]. Trujillo and Olague [14] found that an effective IP operator requires information pertaining to the rate of

change in image intensity values, based on previous research on the analytical properties of corner detectors described in [2, 11], Consequently, they used a terminal set that included first and second order image derivatives. However, they did not assure that this set is either necessary or sufficient.

Below, various terminal sets are presented and tested to capture the key properties of corners, derived from mathematical information. A terminal set obtained by combining intensity-related information, several mask sizes, and continuity of neighboring pixels is proposed as suitable for this task.

Pattern-based terminals. A pattern mask is easy to derive to detect a corner, because the corner point has a different intensity or there is an intensity change in some direction to its neighboring pixels. The following terminal set based on shape of partial patterns is defined first, as follows in Figure 2. The terminals consist of various patterns and are designed to have symmetry in a 7 by 7 rectangular area. In here, an average intensity value of pixels for given patterns is used as a node of a GP tree.

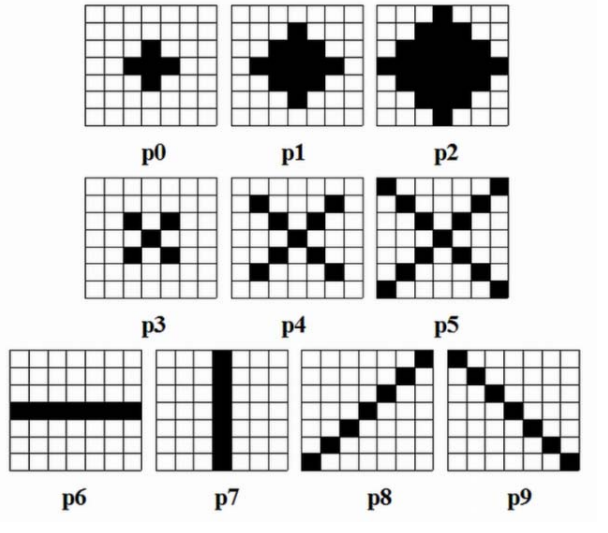


Fig. 2. Pattern-based terminals

Mask area and intensity-based terminals. For each pixel in a rectangular mask, the intensity of the pixel relative to the center pixel is classified into one of darker, similar, or brighter states in Figure 3. Additionally, different sizes of rectangular masks are used 3x3, 5x5, and 7x7. The terminals are defined combining these two intensity and size measures of masks as shown in equation (2). For example, the d_3 represents the number of pixels of darker intensity than a center pixel in a 3x3 mask, and s_7 represents the number of pixels of similar intensity to a center pixel in a 7x7 mask.

$$M_{(i\times i)} = \begin{cases} d_i, & \leq I_n - t \\ s_i, & I_n - t < I_n < I_n + t \\ b_i, & I_n + t \leq I_n \end{cases} \quad (2)$$

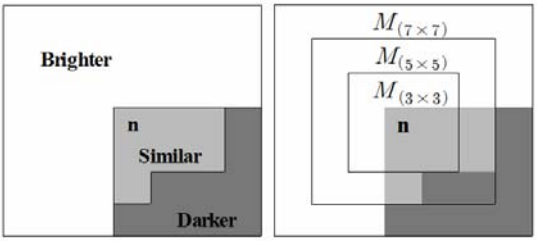


Fig. 3. Terminals based on partitions of mask and brightness

Contiguity and intensity-based terminals. It may not always be possible to classify a pixel as a corner based only on the number of pixels matching certain area- and intensity-based terminals. In order to improve on this, we propose to use the number of pixels of similar intensity in contiguous regions of various sizes as a terminal, as illustrated in Figure 4. Just as was done with the mask area and intensity-based terminals, the contiguity and intensity-based terminals are defined according to different sizes of rectangular masks 3x3, 5x5, and 7x7.

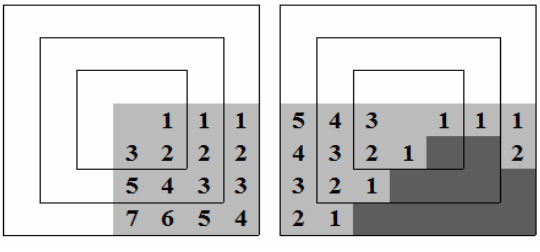


Fig. 4. Terminals based on continuity and intensity

2.4 Fitness Function

The goal of corner detection in our paper is to achieve robust performance under rotation and scaling. First, both MR (Miss Rate not finding a corner) and FPR (False Positive Rate calling a feature a corner that is not) are used as basic indexes below.

The fitness function is based on a combination of the MR and the FPR under scaling and rotational transformations. The coefficient α is a weighing value for the miss rate, for SI_{MR} and RI_{MR}. β is used to weight rotated errors RI_{MR} and RI_{FPR}. SI and RI represent scaled index and rotated index.

The fitness of a genetic program is obtained as follows. Apply the program as a moving *n* x *n* window template (*n* is the size of the input field) to each of the training images and obtain the output value of the program at each possible window position. Match the detected corners with the known locations of each of the desired true corners. Calculate the MR and the FPR of the evolved GP tree. Compute the fitness function as follows in equation (3):

$$MR = (miss/total\_corners) \times 100$$
$$FPR = (false/total\_non\text{-}corners) \times 100$$

$$Fitness\ Fuction = (SI_{MR} \times \alpha + SI_{FPR}) + ((RI_{MR} \times \alpha + RI_{FPR}) \times \beta) \quad (3)$$

3 Experiments and Analysis

3.1 Experimental Environments

We use five images in the experiments (Figure 6 in section 3.4). One is a commonly used synthetic benchmark image and the others are scaled and rotated images) of the original one. The original image includes many corner types, including L-Junctions, Y-Junctions, T-Junctions, Arrow-Junctions and X-Junctions. It has been widely used to evaluate corner detectors. Many corner-like edges are generated when the image is rotated. We have tested various images rotated by 30, 45, and 60 degrees.

The GP programs were run on a Pentium Core2Duo 2.66GHz with 4GB RAM using GPLAB ver.3. [12] The GP parameters used for the corner detector design were as follows:

Number of generations: 100
Population sizes: 100, 200, 300
Max depth: 7, 10, 12, 17
Selection: Tournament (size=7)
Crossover: 0.8
Mutation: 0.2

3.2 Experiments with Various Terminal Sets

First, experiments were executed on the original image using various terminal sets. We tested and compared the three terminal sets defined in Section 2: pattern-based terminals (T_1), mask area and intensity-based terminals (T_2), and contiguity and intensity-based terminals (T_3). T_1 and T_2 were used independently, but T_3 was used in combination with T_2.

The comparison results in Table 1 show that terminal set T2 + T3 was much better on this problem than were T1 or T2. The reason may be that this terminal set was carefully designed for robustness, combining intensity-related information, several mask sizes, and continuity of neighboring pixels. Therefore, terminal set T2 + T3 was chosen for use in the experiments that follow.

Table 1. Comparison of error rates on original image for various terminal sets

Terminal	*MR(%)*	*FPR(%)*	*MR + FPR*
T_1	1.53	8.48	10.01
T_2	3.84	1.89	5.73
$T_2 + T_3$	0	1.23	1.23

3.3 Experiments Using Various Population Size and Tree Depths

Typical GP runs with population sizes of 100, 200 and 300 are compared again run against only the original, unrotated and unscaled image. Ten trials were done for each population size. The average values of error rates for SI_{MR}, SI_{FPR}, RI_{MR}, and RI_{FPR} are displayed in Figure 5(left). Population size 300 shows better results than other sizes, for all indexes. The error rates of SI_{MR} and RI_{MR} are quite sharply reduced.

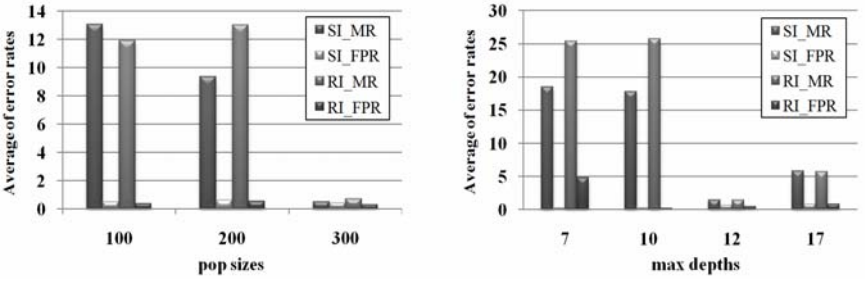


Fig. 5. Error rate indexes by various population sizes and tree depths

The maximum depth of the GP tree is an important factor that affects the structure of the solution, the performance, and the time needed for evolution of a good solution. Various maximum tree depths, including 7, 10, 12 and 17, were tested with population size 300 on the given images. The average values of error rates for SI_{MR}, SI_{FPR}, RI_{MR}, and RI_{FPR} are displayed versus tree depth in Figure 5(right). GP tree max depth of 12 showed the best results. For the case of max depth 17, it is speculated that there might have been a problem such as proliferation of introns that hindered the search efficiency of the evolution process.

3.4 Experimental Comparison with Existing Methods

Below, the proposed GP-based corner detector is compared to several typical corner detectors, namely Harris, SUSAN, and FAST, on all five test images. Table 2 shows the results of detected corners of scaled (80%) and rotated (30°, 45°, 60°) images for four methods: Harris, SUSAN, FAST, and the proposed GP-based method, where N_{ground} is the total number of corners in the ground truth data, $N_{detected}$ is the total number of corners detected by a specific corner detector, and $N_{correct}$ is the total number of correct matching corners when comparing the ground truth data.

Table 2. Comparison of the number of detected corners

		N_{ground}	$N_{detected}$	$N_{correct}$
Original	**Harris**	78	92	75
	SUSAN		83	76
	FAST		75	75
	GP		80	78
80%	**Harris**	78	87	73
	SUSAN		90	75
	FAST		83	75
	GP		79	76
30°	**Harris**	78	133	73
	SUSAN		113	77
	FAST		84	76
	GP		87	78
45°	**Harris**	78	97	76
	SUSAN		120	76
	FAST		84	75
	GP		80	77
60°	**Harris**	78	132	71
	SUSAN		115	76
	FAST		85	77
	GP		87	78

The detected corners of Harris and SUSAN for all five images had bigger differences from the total number of real corners than FAST and the proposed GP-based method had, especially in cases of rotated (30°, 45°, 60°) images. Between FAST and GP-based method, the detected corners of GP were slightly closer to the number of real corners.

In terms of the total number of correctly matched corners, GP-based method was better than FAST, and much better than Harris and SUSAN across all images. The proposed detector clearly outperformed the others. Therefore it is verified that the proposed detectors' performance was superior to existing detectors on the scaled and rotated transformed images tested.

Figure 6 shows images of corner maps obtained by the Harris, SUSAN, FAST, GP-based corner detectors. We can see that Harris and SUSAN detectors mis-detect as corners many corner-like edges in rotated images, especially as shown in the first and second columns of Figure 6. The FAST detector reduced sharply the false detection of these corner-like edges, but missed several actual corners in rotated images. The GP-based corner detector shows the best performance in both false positives and false negatives as shown on the far right of Figure 6.

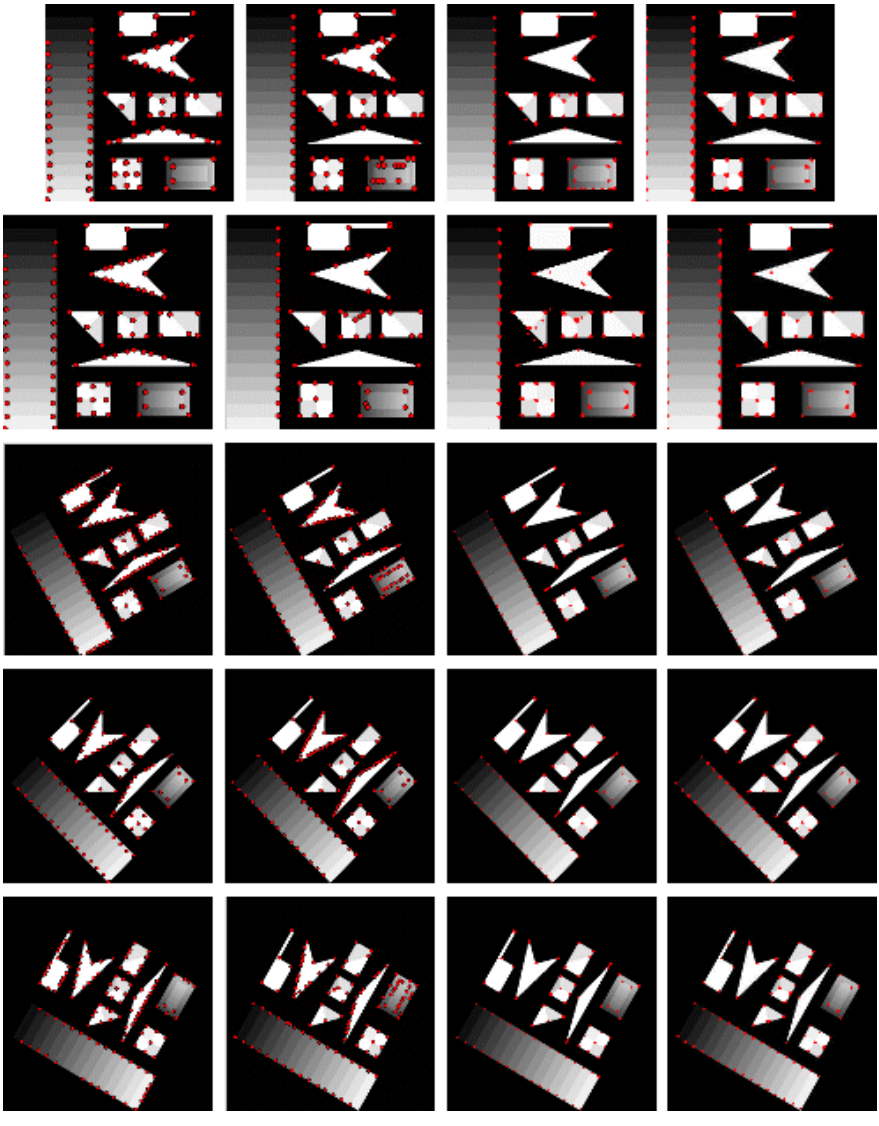


Fig. 6. Result images of corner maps (80%, 100%, 30°, 45°, 60° transformed images from top, and Harris, SUSAN, FAST, GP-based corner detector form left)

4 Conclusions

GP- (Genetic Programming-) based robust corner detectors for scaled and rotated images are presented. Our approach posed corner detection as an optimization problem, and developed an evolution methodology which enables automatic generation of corner detectors. GP terminals and functions were carefully designed to discriminate

corners correctly from noise and corner-like edges. Various terminal sets were presented and tested to capture the key properties of corners, derived from mathematical properties. Combining intensity-related information, several mask sizes, and continuity of neighboring pixels allowed generation of a high-performance terminal set. The fitness function was defined to promote performance in terms of both accuracy and robustness for scaled and rotated images.

The proposed GP-based corner detector is compared to three typical existing corner detectors, Harris, SUSAN and FAST, on a test image and its scaled and rotated versions. Performance measures were selected to evaluate accuracy and robustness for scaled and rotated images. Experimental results showed that the proposed approach generated a superior corner detector to those of existing methods. Further study will aim at refinement of the GP representation, improvement of the GP evolution, seeking of image-independent generalizations about terminal sets, and application to more complex natural images.

Acknowledgments

This work was supported by National Research Foundation of Korea Grant funded by the Korea government (2009-0071419).

References

1. Besetti, S., Soule, T.: Function Choice, Resiliency and Growth in Genetic Programming. In: Proceedings of the 2005 conference on Genetic and Evolutionary Computation (GECCO 2005), USA, pp. 1771–1772 (2005)
2. Bastanlar, Y., Yardimci, Y.: Corner Validation based on Extracted Corner Properties. Computer Vision and Image Understanding 112(3), 243–261 (2008)
3. Ebner, M.: On the Evolution of Interest Operators using Genetic Programming. In: Proceeding of the First European Workshops on Genetic Programming (EuroGP 1998), Paris, pp. 6–10 (1998)
4. Edner, M., Zell, A.: Evolving Task Specific Image Operator. In: Proceeding of the First European Workshops on Evolutionary Image Analysis, Signal Processing and Telecommunications (EvolASP 1999), Goteborg, pp. 74–89 (1999)
5. Harris, C., Stephens, M.: A Combined Corner and Edge Detector. In: Proceeding of the 4th Alvey Vision Conference, UK, pp. 147–151 (1988)
6. Kitshen, L., Rosenfeld, A.: Gray-level corner detection. Pattern Recognition Letters 1(2), 95–102 (1982)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge (1992)
8. Moravec, H.P.: Visual mapping by a robot rover. In: Proceeding of the 6th International Joint Conference on Artificial Intelligence (IJACI), Tokyo, pp. 598–600 (1979)
9. Olague, G., Hernández, B.: A New Accurate and Flexible Model based Multi-Corner Detector for Measurement and Recognition. Pattern Recognition Letters 26(1), 27–41 (2005)
10. Rosten, E., Porter, R., Drummond, T.: Faster and better: a machine learning approach to corner detection. IEEE Trans. Pattern Analysis and Machine Intelligence (2008)

11. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of Interest Point. International Journal of Computer Vision 37(2), 151–172 (2000)
12. Silva, S.: GPLAB: A Genetic Programming Toolbox for MATLAB. Version 3 (2009), http://gplab.sourceforge.net/index.html
13. Smith, S.M., Brady, J.B.: SUSAN-A New Approach to Low Level Image Processing. International Journal of Computer Vision 23(1), 45–78 (1997)
14. Trujillo, L., Olague, G.: Synthesis of Interest Point Detectors through Genetic Programming. In: Proceeding of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO 2006), Seattle, pp. 887–893 (2006)
15. Zhang, M.: Improving Object Detection Performance with Genetic Programming. International Journal on Artificial Intelligence Tools 16(5), 849–873 (2007)
16. Zhang, M., Gao, X., Lou, W.: A New Crossover Operator in Genetic Programming for Object Classification. IEEE Trans. Systems, Man and Cybernetics, Part B 37(5), 1332–1343 (2007)

Self-organized and Evolvable Cognitive Architecture for Intelligent Agents and Multi-Agent Systems

Oscar Javier Romero López

Fundación Universitaria Konrad Lorenz, Bogotá, Colombia
ojrlopez@hotmail.com

Abstract. Integrating different kinds of micro-theories of cognition in intelligent systems when a huge amount of variables are changing continuously, with increasing complexity, is a very exhaustive and complicated task. Our approach proposes a hybrid cognitive architecture that relies on the integration of emergent and cognitivist approaches using evolutionary strategies, in order to combine implicit and explicit knowledge representations necessary to develop cognitive skills. The proposed architecture includes a cognitive level controlled by autopoietic machines and artificial immune systems based on genetic algorithms, giving it a significant degree of plasticity. Furthermore, we propose an attention module which includes an evolutionary programming mechanism in charge of orchestrating the hierarchical relations among specialized behaviors, taking into consideration the global workspace theory for consciousness. Additionally, a co-evolutionary mechanism is proposed to propagate knowledge among cognitive agents on the basis of memetic engineering. As a result, several properties of self-organization and adaptability emerged when the proposed architecture was tested in an animat environment, using a multi-agent platform.

Keywords: Cognitive architectures, gene expression programming, artificial immune systems, neural nets, memetics.

1 Introduction

In the last fifty years, the study of artificial cognitive systems have involved a number of disciplines such as artificial intelligence, cognitive science, psychology and more, in order to determine the necessary, sufficient and optimal conditions and resources for the development of agents exhibiting emergent intelligence. There are several theories of cognition, each taking a different position on the nature of cognition, what a cognitive system should do, and how a cognitive system should be analyzed and synthesized. From these, it is possible to discern three broad classes: the cognitivist approach based on symbolic information processing representational systems; the emergent systems approach embracing connectionist systems, dynamical systems, and enactive systems, all based on a lesser or greater extent of principles of self-organization [1],[2]; and the hybrid approach which combine the best of the emergent systems and cognitivist systems [3].

Some of the most relevant cognitive architectures which follow a cognitivist approach are: SOAR [4], ACT-R [5], ICARUS [3], and EPIC [3]. Some of the architectures of the

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 392–401, 2010.

emergent approach of major importance are: GW [6], SASE [3], and DARWIN [3]. The hybrid approach architectures are known as CEREBUS [3], KISMET [7], CLARION [8], Polyscheme[9], and LIDA[10]. Some of these architectures deal with aspects of cognitive modeling and representation; some others include learning modules, inference and knowledge generalization; and there are others that try to go further and add motivational and meta-cognition components. The hybrid approach is more complex and of greater interest to us since it seeks to unify the different dichotomies of symbolic vs. sub-symbolic models, explicit vs. implicit learning, and cognitive vs. emergent approaches. However, a common weakness in the hybrid approach architectures is that they usually abridge the system functionality into a rigid structure of symbolic and sub-symbolic components resulting in a poor ability to self-organize and adapt to new environments.

The present research focuses on implementing a hybrid architecture for cognitive agents supported by both cognitivist and emergent approaches. On the one hand, the cognitivist approach provides an explicit knowledge representation through the use of symbolic AI techniques. On the other hand, the emergent approach defines three evolutionary strategies as observed in nature [11]: Epigenesis, Ontogenesis, and Phylogenesis, endowing the architecture with implicit knowledge learning, sub-symbolic representations, and emergent behavior guided by bio-inspired computational intelligence techniques. As the cognitivist approach is well known, we will briefly describe it here before elaborating on the emergent approach. The remainder of this paper is organized as follows. The description of the proposed architecture is detailed in Section 2. Sections 3, 4, and 5 describe in more detail each module of the emergent approach according to the three evolutionary strategies. Section 6 outlines and discusses the results of the experiments. The concluding remarks are shown in Section 7.

2 Proposed Hybrid Cognitive Architecture

Figure 1 gives an overview of the hybrid architecture which has six main modules: Attention module, Procedural module, Intentional/Declarative module, Motor module, Motivational module, and Co-evolutionary module. Each module is composed of sub-modules with more specific functionalities which are communicated to each other by broadcasting mechanisms. Architecture is distributed in two dimensions: horizontal and vertical dimensions. At horizontal dimension, modules belong to either emergent or cognitivist level, whereas modules at vertical dimension are distributed according to their functionality (attention, procedural reasoning, intentions, motor processing etc.).

We will first give a brief description of all the modules of the architecture and then provide a more detailed description of those modules that have been developed so far. Initially, our work has focused on developing the procedural and co-evolutionary modules and their interaction with attention and motor modules. The remainder of the modules will be considered in subsequent stages of the research, and therefore are not described in this work.

The Procedural module corresponds to an area of the mammalian brain called Basal Ganglia [5] which is in charge of functions such as rule matching, conflict resolution, cognition, learning, and selection and the execution of actions. This module is composed of several components called Specialist Behaviors (SB), which are organised horizontally, and three sub-modules which are distributed vertically. The three

sub-modules are: the connectionist module, the autopoietic machines module, and the productions module. The horizontally-formed components of the procedural module will be explained in the next section. The Connectionist module (found at the emergent level of the diagram) models innate skills, which require less processing time in comparison with other deliberation processes. It therefore uses the Backpropagation Neural Networks (BNN) which is more appropriate for enacting reactive reasoning. The Autopoietic Machines module is formed by multiple self-organized and self-regulated systems: Artificial Immune Systems [17], where each one models a set of sub-symbolic rules on the basis of autopoietic principles [11]. The Productions module manages different sets of symbolic rules which simulates either the innate knowledge passed on genetically by an evolutionary process, or the knowledge acquired by a previous experience.

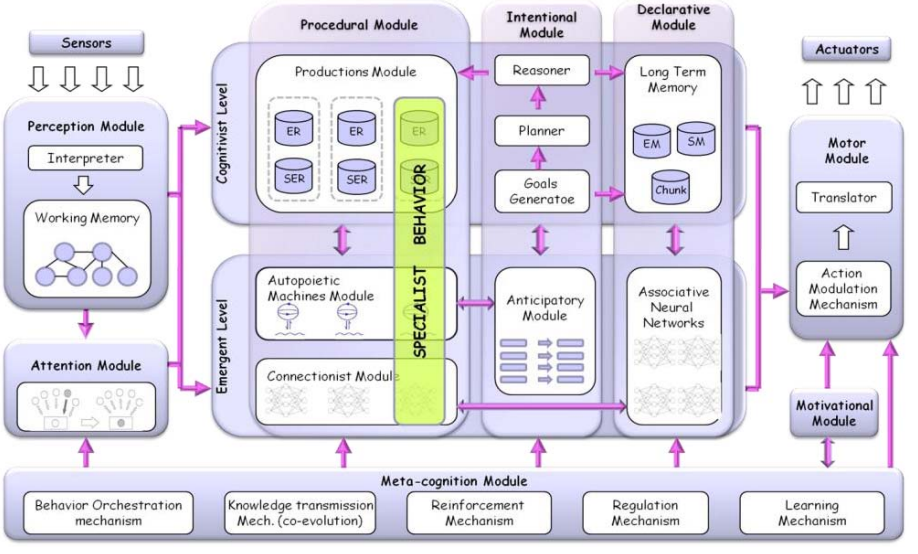


Fig. 1. Hybrid Cognitive Architecture

The Intentional module represents the agent's intentionality through goal and plan generation at the cognitivist level, as well as prospection strategies and internal simulation at the emergent level. This module will have a declarative knowledge representation composed of chunks of semantic and episodic memories which are accessed indirectly, as proposed in [8]. This module is able to predict the outcomes of the actions produced by the system and construct a model of events for controlling perception through stimuli anticipation.

The Attention module has the responsibility of interpreting the perceived information through the sensors and transforming it into percepts (sensory inputs translated into property/value pairs). The Attention module is based on Global Workspace theory and Theater Metaphor for Consciousness [6]. This module has the responsibility for coordinating the execution of several SBs, which compete and cooperate in order to get the attention focus (consciousness). The most innovative aspect of this module

is the behavior orchestration managed by a mechanism that uses Gene Expression Programming (GEP), an evolutionary programming algorithm proposed by Ferreira [12]. This mechanism will be discussed later in section 4.

3 Epigenetic Approach: Implicit Learning at Emergent Level

The epigenesis refers to heritable changes in phenotype (appearance) or gene expression, caused by mechanisms other than changes in the underlying DNA sequence. Therefore, the epigenesis represents the final tuning process by means of each individual adapts efficiently to its environment from the abilities included in its genetic code.

In our work, the epigenetic approach references to the mechanisms that allow agent modifying some aspects of its both internal and external structure as a result of interacting with its environment, in other words, "learning". Therefore, we propose the development of two main approaches which intend to simulate the most evident epigenetic systems observed in nature: the central nervous system (CNS) and the immune system. In our work, the connectionist module which represents the CNS is organized in groups of Backpropagation Neural Networks (BNN), each one representing the sub-symbolic specialization of a task as in [13]. Each specialized BNN is classified according to its purpose, in order to filter the perceived stimuli from environment and select the respective reactive actions. We propose an AIS as autopoietic machine [11] which starts with an sensory input data set (antigens) that stimulate an immune network, and then goes through a dynamic process until it reaches some type of stability. Specifically, each autopoietic machine is based on AiNet [15], a model which implements a discrete immune network that has been developed for data compression and clustering and later for optimization.

3.1 Vertical Integration: Specialist Behaviors

The Specialist Behaviors (SB) are proposed as hybrid units of procedural processing which are in charge of specializing the cognitive skills of the architecture. These specialists are hybrid because of incorporation of both symbolic and sub-symbolic elements at the procedural module. In particular, the procedural module arranges a set of SBs distributed vertically, every one made up of each horizontally-formed component (i.e., an specific SB has one BNN, one AIS, one ER set, and one SER set, as in Figure. 1).

Thus, SBs help the architecture to articulate the set of skills because of each SB attends on an specific set of stimuli signals and gives an appropriated response to the environment. Accordingly, each SB can be formalized as follows:

$$<SB> = <ER> \cup <SER> \cup <AM> \cup <BNN>$$

The purpose of including multiple components in an SB is that each one compensates the weaknesses of the rest. For example, BNN are often better at making forward inferences about object categories than ERs, whereas ERs are often better at making forward inferences involving causal change than neural networks. AIS is able to make both kind of inferences from implicit representations but it involves more processing

time discovering new rules than the other two components. In addition, a reinforcement signal (as a factor of learning in the procedural module) is used to modify the future responses of the agent. This is achieved through adjusting the connections in BNNs, rewarding the activated antibodies in AISs, and extracting sub-symbolic knowledge from emergent level in SERs.

4 Ontogenetic Approach: Behavior Orchestration

Ontogenetic principles involve all the mechanisms in charge of developing an agent on the basis of the stored information in its own genetic code without interposing the environment influence. Additionally, it defines the history of structural change in a unity without the lost of organization that allows that unity to exist. Some outcomes of these principles as self-replication and self-regulation properties in biological systems can be valued. In our work, the ontogenetic approach is simulated through the interaction among different modules: the Attention module, the Goal module, the Anticipatory Module, and the SBs in Procedural module. The main idea in this approach is that the attention module supported by Global Workspace theory, orchestrates the different SBs in such a way that either cooperate or compete among them.

The attention module defines a set of attention machines (AM), which are systems implemented as attention fixations that execute algorithms by sequences of SBs. Each AM has a set of premises that describe the pre-conditions of AM activation, the stream of SBs, and the post-conditions that will have to guarantee after the execution of the stream. Te pre-conditions indicate the goals, expectations, emotions, and stimuli (provided by the working memory) which the agent will have to process and satisfy at any given time. The stream of SBs is a sequence of SBs and relations among them which describes how the agent must behave in a particular situation. Finally, post-conditions are a set of states and new goals generated after the execution of the stream. The main question that addresses the development of the attention module is how it will be able to arbitrate autonomously the execution of SBs in each AM given a set of stimuli, expectations, goals, and emotions?

As a result, we propose an evolutionary model based on GEP [12] that is used to evolve the AMs in order to generate an appropriated behavior orchestration without defining a priori the conditions of activation about each SB. GEP uses two sets: a function set and a terminal set. Our proposed function set is: IFMATCH, AND, OR, NOT, INHIBIT, SUPRESS, AGGREGATE, COALITION, and SUBORDINATION. The AND, OR and NOT functions are logic operators used to group or exclude subsets of elements (SBs, goals, working memory items, etc.). The conditional function IFMATCH is an applicability predicate that matches specific stimuli. This function has three arguments; the first argument is the rule's antecedent, an eligibility condition which correspond with a subset of sensory inputs, motivational indicators (internal states, moods, drives, etc.), and working memory elements, which model the agent's current state. All elements of these subsets are connected with logic operators. If the whole set of conditions exceeds a threshold, then the second argument, the rule's consequent, is executed, otherwise the third argument is executed. Second and third argument should be a set of functions such as INHIBIT, SUPPRESS, AGGREGATE, COALITION, or SUBORDINATION, or maybe an AND/OR function connecting more elements when is necessary.

The INHIBIT, SUPPRESS and AGGREGATE functions have two SBs as arguments (SB_A, SB_B) and indicate that SBA inhibits, suppresses, or aggregates SB_B. The COALITION/SUBORDINATION functions, instead of binomial functions mentioned above, perform a set of SBs. The COALITION function describes a cooperation relationship among SBs where actuators may activate multiple actions. The SUBORDINATION function defines a hierarchical composition of SBs which are activated in a specific sequence. In addition to, the terminal set is composed by the SB set, the motivational indicators, the goal set, and the working memory elements. Additionally "do not care" elements are included so whichever SB, motivational indicator, goal, or working memory item can be referenced.

Each agent has a multigenic chromosome, that means, each chromosome has a gene set where each gene is an eligibility rule like in the example, so the agent has several rules (genes) as part of its genotype and each one is applied according to the situation that matches the rule antecedent. Each gene becomes to a tree representation and afterwards some genetic operators are applied among genes of the same agent and genes of other agents, as in [12]. Some of these genetic operators are: selection, mutation, root transposition, gene transposition, two-point recombination and gene recombination, in order to evolve chromosomal information. After certain number of evolutionary generations, valid and better adapted AMs are generated. A roulette-wheel method is used to select individuals with most selection probability derived from its own fitness. Fitness represents how good interaction with environment during agent's lifetime was.

5 Phylogenetic Approach: C-evolutionary Mechanism

In biology, phylogenesis (evolution) collects all those mechanisms which, leaded by natural selection, have given place to the broad variety of species observed in nature. Evolutionary mechanism operates in populations and as a result, it gets a genetic code which allows individuals of a concrete population to adapt to the environment where they live in. On the basis of phylogenetic theory [11], a co-evolutionary mechanism is proposed to evolve fine-grained units of knowledge through the multi-agent system, taking the foundation of meme and memetic algorithms. The term "meme" was introduced and defined by Dawkins [16], as the basic unit of cultural transmission or imitation that may be considered to be passed on by non-genetic means. In our work, each meme contains a symbolic and sub-symbolic representation of knowledge, and also a set of indicators such as demotion, reliability, rule support and fitness.

As a result, our co-evolutionary mechanism is based on a Memetic Algorithm [16] which is inspired by both Darwinian principles of natural evolution and Dawkins' notion of a meme. This mechanism can be viewed as a population-based hybrid genetic algorithm coupled with an individual learning procedure capable of performing local refinements. Most evolutionary approaches use a single population where evolution is performed; instead of this, in our co-evolutionary approach, the SBs are discriminated in categories and make them evolve in separate pools without any interaction among themselves. After certain period of time a co-evolutionary mechanism is activated. For each behavior pool, a stochastic selection method is executed, where those SBs that had the best performance (fitness) will have more probability to reproduce. Then, a crossover genetic operator is applied among each pair of selected SBs and some memes are both selected and interchanged with other ones.

6 Experimentation

In order to evaluate the proposed cognitive model, following aspects were considered:

- Learning convergence and knowledge generalization of the procedural module,
- Analysis of eligibility rules evolved by GEP in the attention module, and
- Learning convergence of the co-evolutionary mechanism.

An artificial life environment called Animat (animal + robot) is proposed to test the experiments. The environment simulates virtual agents competing for getting food and water, avoiding obstacles, and so forth. Each animat, driven by an agent, disposes a set of 8 proximity sensors around itself. In Figure 2 is depicted the three environments and the desired paths that the agent should cross. The environment in Figure 2a. was a basic learning scenario where the agent had to follow a simple path. In Figure 2b. some little changes were included, and in Figure 2c. the environment was more complex because of a new path of food and more elements that were introduced. The experiments were made using three agent behaviors: looking for food (SB-eat), avoiding obstacles (SB-avoid-obstacles), and escaping from predators (SB-escaping-from-predators).

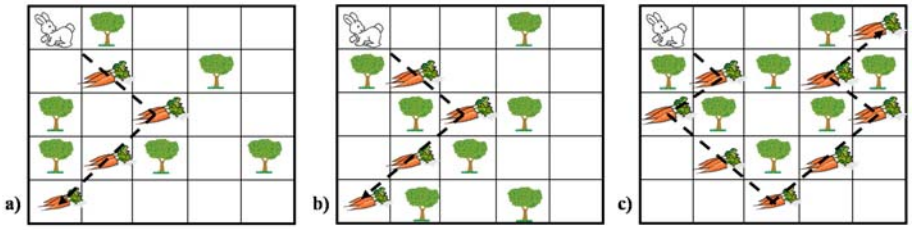


Fig. 2. Animat environments. a) basic environment, b) modified environment, and c) complex environment.

Thus, some experiments designed to evaluate the performance aspects mentioned above are described next.

6.1 Analysis of Eligibility Rules Evolved by GEP

After the attention machines in the attention module have evolved during a specific number of generations, we analyze the final eligibility rules of the best adapted agents where emergent properties arose.

Initially, we present an initial eligibility rule which has syntax conflicts; therefore an evolved eligibility rule syntactically well-formed emerges from GEP.

We have chosen an eligibility rule from a non-trained agent and afterwards we show the same evolved eligibility rule but now it has no syntax conflicts and also it's better well-suited than its predecessor.

Eligibility Rule at generation 0:

```
IFMATCH:
   {food},{tree},{empty},{empty},{empty},{empty},
   {empty},{tree} AND {goal-is-eat}
THEN:
 {SB-eat} INHIBITS {SB-avoid-obstacles} AND
 {SB-avoid-obstacles} SUPRESSES {SB-eat}
ELSE:
 SUBORDINATION {SB-avoid-obstacles} AND
   {SB-eat}
```

The above eligibility rule means that when the agent senses "food" around it, it must do something to get the food while is avoiding obstacles, but is contradictory because {SB-eat} can't inhibit {SB-avoid-obstacles} while {SB-avoid-obstacles} is suppressing {SB-eat} at the same time. So, the evolved consequent of the eligibility rule after 17 epochs is:

```
IFMATCH:
   {food},{tree},{empty},{empty},{empty},{empty},
   {empty},{tree} AND {goal-is-eat}
THEN
   COALITION {SB-eat} AND {SB-avoid-obstacles}
ELSE
   {SB-avoid-obstacles} INHIBITS {SB-eat}
```

It is important to notice that evolved eligibility rule does not present any syntax conflict and is a valid rule which forms a coalition among {SB-avoid-obstacles} and {SB-eat} behaviors when the agent reads food and obstacles around it. Otherwise, the agent always will execute the rule: {SB-avoid-obstacles} inhibits {SB-eat}, focusing the agent attention on obstacles because of {SB-eat} behavior has a lower priority and is less reactive than {SB-avoid-obstacles} behavior.

6.2 Learning Convergence of the Co-evolutionary Mechanism

This experiment examines if the fitness of every separate behavior pool increments gradually until it reaches a convergence point while evolution takes place. The experiment was carried out with the parameters on Table 1.

Table 1. Co-evolution learning parameters

Parameter	Value
Epochs	50
Number of epochs per run	50
Crossover probability	0.7
Mutation probability	0.3
Mutation rate η	0.85
Mutation rate θ	0.25
Mutation rate κ	1.03
Mutation rate γ	0.01

Three behavior pools were selected for the experiment: avoiding-obstacles, looking-for-food, and escaping-from-predators. The results are depicted in Figure 3.

Figure 3 depicted some differences in each learning curve, because of environmental conditions, however the pools always tried to converge and reach certain knowledge stability at the same number of epochs (approximately after 30 epochs), that means the evolution has been effective and each behavior pool has established a coherent knowledge base getting a consensus among its own behavior instances and about what the "behavior category" should do.

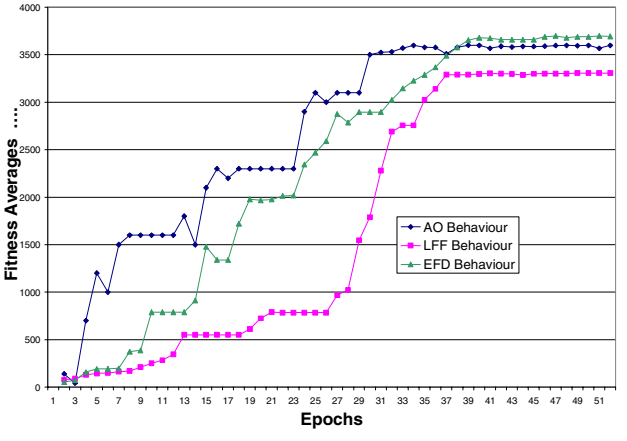


Fig. 3. Evolution convergence rate in 3 behaviour pools

7 Conclusions

The evolutionary mechanisms used in this work, provided a plasticity feature allowing the agent to self-configure its own underlying architecture; thus, it can avoid creating exhaustive and extensive knowledge bases, pre-wired behavior structures of behaviors, and pre-constrained environments. Instead of this, the cognitive agents which use our architecture only need to interact with an arbitrary environment to adapt to it and take decisions in both a reactive and deliberative fashion.

In the experimentation, the emergent properties were difficult to discover because it took a lot of time to evolve the overall system despite of using a multi-agent platform with a distributed configuration. Maybe, it can be similar to the natural evolution where adaptation occurs slowly and sometimes produces poor adapted creatures.

In our future work we expect to continue working on designing more adaptive and self-configurable architectures, incorporating intentional and meta-cognition modules. One concrete application of this research will be the development of a cognitive module for Emotive Pedagogical Agents where the agent will be able to self-learn of perspectives, believes, desires, intentions, emotions and perceptions about itself and other agents, using the proposed approach which will be responsible of driving the cognitive architecture.

References

1. Anderson, M.L.: Embodied cognition: A field guide. Artificial Intelligence 149(1), 91–130 (2003)
2. Berthoz, A.: The Brain's Sense of Movement. Harvard Univ. Press, Cambridge (2000)
3. Vernon, D., Metta, G., Sandini, G.: A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. IEEE Trans. Evolutionary Computation 11(2) (April 2007)
4. Rosenbloom, P., Laird, J., Newell, A. (eds.): The Soar Papers: Research on integrated intell. MIT Press, Cambridge (1993)
5. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psy. Rev. 111(4), 1036–1060 (2004)
6. Shanahan, M.P., Baars, B.: Applying global workspace theory to the frame problem. Cognition 98(2), 157–176 (2005)
7. Breazeal: Emotion and sociable humanoid robots. Int. J. Human-Computer Studies 59, 119–155 (2003)
8. Sun, R., Merrill, E., Peterson, T.: From implicit skills to explicit knowledge: A bottom-up model of skill learning. Cognitive Science 25, 203–244 (2001)
9. Cassimatis, N.L.: Adaptive Algorithmic Hybrids for Human-Level Artificial Intelligence. Advances in Arti. Intell. (2007)
10. Franklin, S.: The LIDA architecture: Adding new modes of learning to an intelligent, autonomous, software agent. In: Proc. of the Int. Conf. on Integrated Design and Process Technology (2006)
11. Maturana, H.R., Varela, F.J.: Autopoiesis and Cognition: The Realization of the Living. Boston Studies on the Philosophy of Science. D. Reidel Publishing Company, Dordrecht (1980)
12. Ferreira, C.: Gene Expression Programming: A new adaptive algorithm for solving problems. Complex Systems 13, 87–129 (2001)
13. Rumelhart, D., McClelland, J.: Parallel Distributed Processing: Explorations in the Microstructures of Cognition. MIT Press, Cambridge (1986)
14. Watkins, C., Dayan, P.: Q-learning. Machine Learning 3, 279–292 (1992)
15. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach (2002)
16. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1989)
17. Romero, D., Niño, L.: An Immune-based Multilayered Cognitive Model for Autonomous Navigation, CEC, Vancouver, pp. 1115–1122 (2006)

Investigating the Local-Meta-Model CMA-ES for Large Population Sizes

Zyed Bouzarkouna[1,2], Anne Auger[2], and Didier Yu Ding[1]

[1] IFP (Institut Français du Pétrole)
1,4 avenue de bois préau
92852 Rueil-Malmaison Cedex, France
[2] TAO Team, INRIA Saclay-Ile-de-France
LRI, Paris Sud University
91405 Orsay Cedex, France

Abstract. For many real-life engineering optimization problems, the cost of one objective function evaluation can take several minutes or hours. In this context, a popular approach to reduce the number of function evaluations consists in building a (meta-)model of the function to be optimized using the points explored during the optimization process and replacing some (true) function evaluations by the function values given by the meta-model. In this paper, the local-meta-model CMA-ES (lmm-CMA) proposed by Kern et al. in 2006 coupling local quadratic meta-models with the Covariance Matrix Adaptation Evolution Strategy is investigated. The scaling of the algorithm with respect to the population size is analyzed and limitations of the approach for population sizes larger than the default one are shown. A new variant for deciding when the meta-model is accepted is proposed. The choice of the recombination type is also investigated to conclude that the weighted recombination is the most appropriate. Finally, this paper illustrates the influence of the different initial parameters on the convergence of the algorithm for multimodal functions.

Keywords: Optimization, Covariance Matrix Adaptation, Evolution Strategy, CMA-ES, Meta-models.

1 Introduction

Many real-world optimization problems are formulated in a black-box scenario where the objective function to optimize $f : \mathbb{R}^n \mapsto \mathbb{R}$ may have noise, multiple optima and can be computationally expensive. Evolutionary algorithms (EAs) are stochastic population based optimization algorithms that are usually a good choice to cope with noise and multiple optima. For expensive objective functions–several minutes to several hours for one evaluation–a strategy is to couple EAs with meta-models: a model of f is built, based on "true" evaluations of f, and used during the optimization process to save evaluations of the expensive objective function [1]. One key issue when coupling EAs and meta-models is to decide when the quality of the model is good enough to continue exploiting

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 402–411, 2010.

this model and when new evaluations on the "true" objective functions should be performed. Indeed, performing too few evaluations on the original objective function can result in suboptimal solutions whereas performing too many of them can lead to a non efficient approach.

The covariance matrix adaptation ES (CMA-ES) [2,3] is an EA recognized as one of the most powerful derivative-free optimizers for continuous optimization[1]. At each iteration of CMA-ES, the evaluation of candidate solutions on the objective function are performed. However, from those evaluations only the ranking information is used. In consequence the algorithm is invariant to transformations on f preserving the ranking. CMA-ES was coupled with local meta-models by Kern et al. [4]. In the proposed algorithm, lmm-CMA, the quality of the meta-model is appraised by tracking the change in the *exact ranking* of the best individuals. The lmm-CMA algorithm has been evaluated on test functions using the default population size of CMA-ES for unimodal functions and for some multimodal functions and has been shown to improve CMA-ES [4].

In this paper, we analyze the performance of lmm-CMA when using population sizes larger than the default one. We show that tracking the exact rank-change of the best solutions to determine when to re-evaluate new solutions is a too conservative criterion and leads to a decrease of the speedup with respect to CMA-ES when the population size is increased. Instead we propose a less conservative criterion that we evaluate on test functions. This paper is structured as follows. Section 2 gives an overview of CMA-ES and lmm-CMA. In Section 3, we evaluate lmm-CMA-ES for population size larger than the default one. In Sections 4.1 and 4.2, we propose a new variant of lmm-CMA. In Section 4.3, the influence of the recombination type on the new variant is tested. The influence of initial parameters is analyzed in Section 4.4.

2 CMA-ES with Local Meta-Models

The Covariance Matrix Adaptation ES. CMA-ES is a stochastic optimization algorithm where at each iteration g, a population of λ points is sampled according to a multivariate normal distribution. The objective function of the λ points is then evaluated and the parameters of the multivariate normal distribution are updated using the feedback obtained on the objective function. More specifically, let $(\mathbf{m}^g, g \in \mathbb{N})$ be the sequence of mean values of the multivariate normal distribution generated by CMA-ES, constituting the sequence of estimate of the optimum and let $(\sigma^g, g \in \mathbb{N})$ and $(\mathbf{C}^g, g \in \mathbb{N})$ be respectively the sequences of step-sizes and covariance matrices. Assume that $\mathbf{m}^g, \sigma^g, \mathbf{C}^g$ are given, new points or *individuals* are sampled according to:

$$\mathbf{x}_i^g = \mathbf{m}^g + \sigma^g \mathcal{N}_i(0, \mathbf{C}^g), \qquad \text{for } i = 1 \dots \lambda \ , \tag{1}$$

where $(\mathcal{N}_i(0, \mathbf{C}^g))_{1 \le i \le \lambda}$ are λ independent multivariate normal distributions with zero mean vector and covariance matrix $\mathbf{C}^g$. Those λ individuals are ranked according to f:

[1] See http://coco.gforge.inria.fr/doku.php?id=bbob-2009-results

$$f(\mathbf{x}^g_{1:\lambda}) \leq \ldots f(\mathbf{x}^g_{\mu:\lambda}) \leq \ldots f(\mathbf{x}^g_{\lambda:\lambda}) \tag{2}$$

where we use the notation $\mathbf{x}^g_{i:\lambda}$ for i^{th} best individual. The mean $\mathbf{m}^g$ is then updated by taking the weighted mean of the best μ individuals, $\mathbf{m}^{g+1} = \sum_{i=1}^{\mu} \omega_i \mathbf{x}^g_{i:\lambda}$, where in general $\mu = \frac{\lambda}{2}$ and $(w_i)_{1\leq i\leq \mu}$ are strictly positive and normalized weights, i.e., satisfying $\sum_{i=1}^{\mu} \omega_i = 1$. The default weights are equal to:

$$\omega_i = \frac{\ln(\mu+1) - \ln(i)}{\mu \ln(\mu+1) - \ln(\mu!)}, \qquad \text{for } i = 1 \ldots \mu. \tag{3}$$

Furthermore σ^g and $\mathbf{C}^g$ are updated as well after evaluation and we refer to [3] for the equation updates. All updates rely on the ranking determined by Eq. 2 only and not on the exact value of the objective functions such that the CMA-ES is invariant when optimizing f or $g \circ f$ where $g : \mathbb{R} \mapsto \mathbb{R}$ is a strictly increasing mapping. The default population size λ equals $4 + \lfloor 3 \ln(n) \rfloor$.

Locally weighted regression. During the optimization process, a database, i.e., a training set is built by storing, after every evaluation on the true objective function, points together with their objective function values $(\mathbf{x}, y = f(\mathbf{x}))$. We will later then show that some points whose evaluation is asked by the optimizer are not evaluated on the true objective function. Assuming that the training set contains a sufficient number m of couples $(\mathbf{x}, f(\mathbf{x}))$, for each individual in the population, denoted now $\mathbf{q} \in \mathbb{R}^n$, locally weighted regression builds an approximate objective function using (true) evaluations stored in the training set. Kern et al [4] have tested several types of models for the objective function (linear, quadratic, ...) and have investigated the impact of the choice of the model complexity and recommend to use a full quadratic meta-model that we will hence consider in this paper. The full quadratic meta-model is built based on minimizing the following criteria w.r.t. the vector of parameters $\beta \in \mathbb{R}^{\frac{n(n+3)}{2}+1}$ of the meta-model at $\mathbf{q}$:

$$A(\mathbf{q}) = \sum_{j=1}^{m} \left[\left(\hat{f}\left(\mathbf{x}_j, \beta\right) - y_j \right)^2 K\left(\frac{d\left(\mathbf{x}_j, \mathbf{q}\right)}{h} \right) \right] \ , \tag{4}$$

where $\hat{f}$ is the meta-model defined by

$$\hat{f}\left(\mathbf{x}, \beta\right) = \beta^T \left(x_1^2, \cdots, x_n^2, \cdots, x_1 x_2, \cdots, x_{n-1} x_n, x_1, \cdots, x_n, 1\right)^T \ , \tag{5}$$

with $\mathbf{x} = (x_1, \ldots, x_n)$. The kernel weighting function $K(.)$ is defined by $K(\zeta) = (1-\zeta^2)^2 1_{\{\zeta<1\}}$ where $1_{\{\zeta<1\}}$ is one if $\zeta < 1$ and zero otherwise, and d denotes the Mahalanobis distance with respect to the current covariance matrix $\mathbf{C}$ between 2 individuals defined as $d\left(\mathbf{x}_j, \mathbf{q}\right) = \sqrt{\left(\mathbf{x}_j - \mathbf{q}\right)^T \mathbf{C}^{-1} \left(\mathbf{x}_j - \mathbf{q}\right)}$, and h is the bandwidth defined by the distance of the kth nearest neighbor data point to $\mathbf{q}$ where $k = n\left(n+3\right) + 2$.

Table 1. Test functions and their corresponding initial intervals and standard deviations. The starting point is uniformly drawn from the initialized interval.

Name	Function	Init.	σ^0
Noisy Sphere	$f_{\text{NSphere}}(x) = (\sum_{i=1}^{n} x_i^2) \exp(\epsilon \mathcal{N}(0,1))$	$[-3,7]^n$	5
Schwefel	$f_{\text{Schw}}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-10,10]^n$	10
Schwefel$^{1/4}$	$f_{\text{Schw}^{1/4}}(x) = (f_{\text{Schwefel}}(x))^{\frac{1}{4}}$	$[-10,10]^n$	10
Rosenbrock	$f_{\text{Rosen}}(x) = \sum_{i=1}^{n-1} \left(100.\,(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	$[-5,5]^n$	5
Ackley	$f_{\text{Ack}}(x) = 20 - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) + e - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i))$	$[1,30]^n$	14.5
Rastrigin	$f_{\text{Rast}}(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10.\cos(2\pi x_i))$	$[1,5]^n$	2

1 *approximate* $\hat{f}(x_k)$, $k = 1 \ldots \lambda$
2 *rank* the μ best individuals ranking$_0$
3 *evaluate* f for the n_{init} best individuals, add to the training set
4 **for** $n_{ic} := 1$ **to** $\left(\frac{\lambda - n_{init}}{n_b}\right)$ **do**
5 *approximate* $\hat{f}(x_k)$, $k = 1 \ldots \lambda$
6 *rank* the μ best individuals ranking$_{n_{ic}}$
7 **if** (ranking$_{n_{ic}} \neq$ ranking$_{n_{ic}-1}$) **then**
8 *evaluate* f for the n_b best unevaluated individuals, add to the training set
9 **else**
10 **break**
11 **fi**
12 **od**
13 **if** $(n_{ic} > 2)$ **then** $n_{init} = \min(n_{init} + n_b, \lambda - n_b)$
14 **elseif** $(n_{ic} < 2)$ **then** $n_{init} = \max(n_b, n_{init} - n_b)$

Fig. 1. The approximate ranking procedure, performed once the training set contains a sufficient number of evaluations to build the meta-model. n_{init} and n_b are initialized respectively to λ and $\max[1, (\frac{\lambda}{10})]$.

Approximate Ranking Procedure. The lmm-CMA-ES algorithm is using the approximate ranking procedure to decide when the quality of the model built is satisfactory [5]. This procedure heavily exploits the ranked-based property of the CMA-ES algorithm. Fig. 1 gives the implementation of this procedure proposed in [4]. Initially, a number n_{init} of best individuals based on the meta-model are evaluated using the true objective function and then added to the training set. A batch of n_b individuals is evaluated until satisfying the meta-model acceptance criterion: *keeping the ranking of each of the μ best individuals based on the meta-model unchanged for two iteration cycles.* Hence, $(n_{init,g} + n_{ic}\, n_b)$ individuals are evaluated every generation where n_{ic} represents the number of iteration cycles needed to satisfy the meta-model acceptance criterion. We note that n_{init} and n_b are initialized respectively to λ and $\max[1, (\frac{\lambda}{10})]$. The parameter n_{init} is adapted depending on the number of iteration cycles n_{ic}: n_{init} is increased if $(n_{ic} > 2)$ (Line 13 in Fig. 1) and decreased if $(n_{ic} < 2)$ (Line 14 in Fig. 1).

Table 2. Success performance SP1, i.e., the average number of function evaluations for successful runs divided by the ratio of successful runs, standard deviations of the number of function evaluations for successful runs and speedup performance spu, to reach $f_{\text{stop}} = 10^{-10}$ of lmm-CMA and nlmm-CMA. The ratio of successful runs is denoted between brackets if it is < 1.0. Results with a constant dimension $n = 5$ and an increasing λ are highlighted in grey.

Function	n	λ	ϵ	lmm-CMA		spu	nlmm-CMA		spu	CMA-ES	
f_{Rosen}	2	6		291 ± 59		2.7	**252** ± 52		3.1	779 ± 236	
	4	8		776 ± 102	[0.95]	2.8	**719** ± 54	[0.85]	3.0	2185 ± 359	[0.95]
	5	8		1131 ± 143		2.7	**1014** ± 94	[0.90]	3.0	3012 ± 394	[0.90]
	5	16		1703 ± 230	[0.95]	2.0	**901** ± 64		3.7	3319 ± 409	
	5	24		2784 ± 263		1.4	**1272** ± 90	[0.95]	3.0	3840 ± 256	
	5	32		3364 ± 221		1.3	**1567** ± 159		2.9	4515 ± 275	
	5	48		4339 ± 223		1.3	**1973** ± 144		2.9	5714 ± 297	
	5	96		6923 ± 322		1.2	**3218** ± 132		2.5	7992 ± 428	
	8	10		2545 ± 233	[0.95]	2.1	**2234** ± 202	[0.95]	2.4	5245 ± 644	
f_{Schw}	2	6		89 ± 9		4.3	87 ± 7		4.4	385 ± 35	
	4	8		166 ± 8		5.4	166 ± 6		5.4	897 ± 51	
	8	10		334 ± 9		6.2	333 ± 9		6.2	2078 ± 138	
	16	12		899 ± 40		5.9	**855** ± 30		6.2	5305 ± 166	
$f_{\text{Schw}^{1/4}}$	2	6		556 ± 25		2.4	**413** ± 25		3.3	1343 ± 72	
	4	8		1715 ± 87		1.7	**971** ± 36		2.9	2856 ± 135	
	5	8		2145 ± 69		1.6	**1302** ± 31		2.7	3522 ± 136	
	5	16		3775 ± 137		1.3	**1446** ± 31		3.4	4841 ± 127	
	5	24		5034 ± 142		1.2	**1825** ± 45		3.4	6151 ± 252	
	5	32		6397 ± 174		1.2	**2461** ± 43		3.2	7765 ± 227	
	5	48		8233 ± 190		1.2	**3150** ± 58		3.2	10178 ± 202	
	5	96		11810 ± 177		1.2	**4930** ± 94		2.9	14290 ± 252	
	8	10		4046 ± 127		1.5	**2714** ± 41		2.2	5943 ± 133	
f_{NSphere}	2	6	0.35	124 ± 14		2.7	**109** ± 12		3.1	337 ± 34	
	4	8	0.25	316 ± 45		2.3	**236** ± 19		3.1	739 ± 30	
	8	10	0.18	842 ± 77		1.8	**636** ± 33		2.4	1539 ± 69	
	16	12	0.13	2125 ± 72		1.3	2156 ± 216		1.3	2856 ± 88	
f_{Ack}	2	5		302 ± 43	[0.90]	2.6	**227** ± 23		3.5	782 ± 114	[0.95]
	5	7		1036 ± 620		2.0	**704** ± 23	[0.90]	3.0	2104 ± 117	[0.85]
	10	10		2642 ± 93	[0.90]	1.4	**2066** ± 119	[0.95]	1.8	3787 ± 151	[0.95]
f_{Rast}	2	50		898 ± 160	[0.95]	2.7	**524** ± 48	[0.95]	4.7	2440 ± 294	[0.75]
	5	70		19911 ± 599	[0.15]	0.6	**9131** ± 135	[0.15]	1.3	11676 ± 711	[0.50]
	5	140		6543 ± 569	[0.80]	1.6	**4037** ± 209	[0.60]	2.6	10338 ± 1254	[0.85]
	5	280		10851 ± 1008	[0.85]	1.3	**4949** ± 425	[0.85]	2.9	14266 ± 1069	

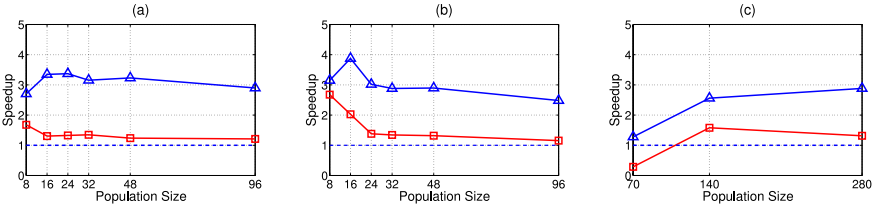


Fig. 2. Speedup of nlmm-CMA (△) and lmm-CMA (□) on (a) $f_{\text{Schw}^{1/4}}$, (b) f_{Rosen} and (c) f_{Rast} for dimension $n = 5$

3 Evaluating lmm-CMA on Increasing Population Size

3.1 Experimental Procedure

The lmm-CMA and the other variants tested are evaluated on the objective functions presented in Table 1 corresponding to the functions used in [4] except two functions: (1) the function $f_{\text{Schw}^{1/4}}$ where we compose the convex quadratic function f_{Schw} by a strictly increasing mapping $g : x \in \mathbb{R} \mapsto x^{1/4}$, introduced because we suspect that the results on f_{Schw} are artificial and only reflect the fact that the model used in lmm-CMA is quadratic and (2) the noisy sphere function

f_{NSphere} whose definition has been modified following the recommendations of [6]. We have followed the experimental procedure in [4] and performed for each test function 20 independent runs using an implementation of lmm-CMA based on a java code of CMA-ES[2] randomly initialized from initial intervals defined in Table 1 and with initial standard deviations σ_0 in Table 1 and other standard parameter settings in [3]. The algorithm performance is measured using the success performance SP1 used in [7]. SP1 is defined as the average number of evaluations for successful runs divided by the ratio of successful runs, where a run is considered as successful if it succeeds in reaching $f_{\text{stop}} = 10^{-10}$. Another performance measure that might be used was the expected running time ERT [8] which is defined as the number of function evaluations conducted in all runs (successful and unsuccessful runs) divided by the ratio of successful runs. In this paper, we opt for SP1 since the stopping criteria for unsuccessful runs were not properly tuned which can affect the performance comparison. We have reproduced the results for the lmm-CMA presented in [4, Table 3]. Those results are presented in Table 2[3].

3.2 Performances of lmm-CMA with Increasing Population Size

In lmm-CMA, a meta-model is accepted if the exact ranking of the μ best individuals remains unchanged. However, this criterion is more and more difficult to satisfy when the population size λ and thus $\mu(= \lambda/2)$ increases. We suspect that this can have drastic consequences on the performances of lmm-CMA. To test our hypothesis we perform tests for $n = 5$ on f_{Rosen}, $f_{\text{Schw}^{1/4}}$ with $\lambda = 8,\ 16,\ 24,\ 32,\ 48,\ 96$ and for f_{Rast} for $\lambda = 70,\ 140,\ 280$. The results are presented in Fig. 2 and in Table 2 (rows highlighted in grey). On f_{Rosen} and $f_{\text{Schw}^{1/4}}$, we observe, as expected that the speedup with respect to CMA-ES drops with increasing λ and is approaching 1. On f_{Rast}, we observe that the speedup for $\lambda = 140$ is larger than for $\lambda = 280$ (respectively equal to 1.6 and 1.3).

4 A New Variant of lmm-CMA

We propose now a new variant of lmm-CMA, the new-local-meta-model CMA-ES (nlmm-CMA) that tackles the problem detected in the previous section.

4.1 A New Meta-Model Acceptance Criteria

We have seen that requiring the preservation of the exact ranking of the μ best individuals is a too conservative criterion for population sizes larger than the default one to measure the quality of meta-models. We therefore propose to

[2] See http : //www.lri.fr/∼hansen/cmaes_inmatlab.html

[3] Experiments have been performed with $k = n(n+3)+2$ indicated in [4]. However we observed some differences on f_{Rosen} and f_{Schw} with this value of k and found out that $k = \frac{n(n+3)}{2} + 1$ allows to obtain the results presented in [4, Table 3]. We did backup this finding by using the matlab code provided by Stefan Kern.

approximate $\hat{f}(x_k)$, $k = 1 \dots \lambda$
determine the μ best individuals set : set_0
determine the best individual : elt_0
evaluate f for the n_{init} best individuals, add to the training set
for $n_{ic} := 1$ **to** $\left(\frac{\lambda - n_{init}}{n_b}\right)$ **do**
  approximate $\hat{f}(x_k)$, $k = 1 \dots \lambda$
  determine the μ best individuals set : $\text{set}_{n_{ic}}$
  determine the best individual : $\text{elt}_{n_{ic}}$
  if $(n_{init} + n_{ic}\, n_b < \frac{\lambda}{4})$
    if $((\text{set}_{n_{ic}} \neq \text{set}_{n_{ic}-1})$ **or** $(\text{elt}_{n_{ic}} \neq \text{elt}_{n_{ic}-1}))$ **then**
      evaluate f for the n_b best unevaluated individuals, add to the training set
    else
      break
    fi
  elseif $(\text{elt}_{n_{ic}} \neq \text{elt}_{n_{ic}-1})$ **then**
    evaluate f for the n_b best unevaluated individuals, add to the training set
  else
    break
  fi
od
if $(n_{ic} > 2)$ **then** $n_{init} = \min(n_{init} + n_b, \lambda - n_b)$
elseif $(n_{ic} < 2)$ **then** $n_{init} = \max(n_b, n_{init} - n_b)$

Fig. 3. The new approximate ranking procedure, performed once the training set contains a sufficient number of evaluations to build the meta-model. n_{init} and n_b are initialized respectively to λ and $\max[1, (\frac{\lambda}{10})]$.

replace this criterion by the following one: after building the model and ranking it, a meta-model is accepted if it succeeds in keeping, both the ensemble of μ individuals and the best individual unchanged. In this case, we ignore any change in the rank of each individual from the best μ individuals, except for the best individual which must be the same, as long as this individual is still an element of the μ best ensemble. Another criterion is added to the acceptance of the meta-model: once more than one fourth of the population is evaluated, the model is accepted if it succeeds to keep the best individual unchanged. The proposed procedure is outlined in Fig. 3. Considering only changes in the whole parent set, without taking into account the exact rank of each individual, and setting an upper limit on the number of true objective function evaluations was first proposed in [9]. The new variant is called nlmm-CMA in the sequel.

4.2 Evaluation of nlmm-CMA

The performance results of nlmm-CMA are presented in Table 2 together with the one of lmm-CMA. Table 2 shows that on f_{Rast}, the nlmm-CMA speedup is in between 2.5 and 5 instead of 1.5 and 3 for lmm-CMA and on f_{Ack}, nlmm-CMA outperforms lmm-CMA with speedups between 1.5 and 3.5 for nlmm-CMA and

Table 3. SP1, standard deviations of the number of function evaluations for successful runs and speedup performance spu, to reach $f_{\text{stop}} = 10^{-10}$ of nlmm-CMA, nlmm-CMA$_I$ (intermediate recombination and default initial parameters), nlmm-CMA$_1$(default recombination, initial values of n_{init} and n_b set to 1) and nlmm-CMA$_2$(default recombination type, $n_{init} = 1$ and $n_b = 1$ during the whole optimization process). The ratio of successful runs is denoted between brackets if it is < 1.0.

Function	n	λ	ϵ	nlmm-CMA	spu	nlmm-CMA$_I$	spu	nlmm-CMA$_1$	spu	nlmm-CMA$_2$	spu
f_{Rosen}	2	6		252 ± 52	3.1	357 ± 67	2.2	250 ± 80	3.1	229 ± 53	3.4
	4	8		719 ± 54 [0.85]	3.0	833 ± 100	2.6	596 ± 55	3.7	575 ± 68	3.8
	8	10		2234 ± 202 [0.95]	2.4	2804 ± 256 [0.95]	1.9	2122 ± 133	2.5	2466 ± 207 [0.85]	2.1
f_{Schw}	2	6		87 ± 7	4.4	110 ± 10	3.5	75 ± 8	5.2	73 ± 7	5.3
	4	8		166 ± 6	5.4	220 ± 15	4.1	138 ± 6	6.5	136 ± 5	6.6
	8	10		333 ± 9	6.2	423 ± 15	4.9	374 ± 16	5.6	380 ± 21	5.5
	16	12		855 ± 30	6.2	947 ± 24	5.6	794 ± 27	6.7	786 ± 37	6.8
$f_{\text{Schw}^{1/4}}$	2	6		413 ± 25	3.3	550 ± 29	2.4	411 ± 20	3.3	398 ± 16	3.4
	4	8		971 ± 36	2.9	1320 ± 76	2.2	938 ± 32	3.1	909 ± 30	3.1
	8	10		2714 ± 41	2.2	2714 ± 257	2.2	2668 ± 40	2.2	2677 ± 36	2.2
f_{NSphere}	2	6	.35	109 ± 12	3.1	135 ± 19	2.5	92 ± 11	3.7	87 ± 9	3.9
	4	8	.25	236 ± 19	3.1	306 ± 40	2.4	216 ± 16	3.4	219 ± 16	3.4
	8	10	.18	636 ± 33	2.4	788 ± 47	2.0	611 ± 35	2.5	619 ± 45	2.5
	16	12	.13	2156 ± 216	1.3	2690 ± 421	1.1	2161 ± 148	1.3	2195 ± 142	1.3
f_{Ack}	2	5		227 ± 23	3.5	329 ± 29 [0.85]	2.4	226 ± 21 [0.95]	3.5	208 ± 19	3.8
	5	7		704 ± 23 [0.90]	3.0	850 ± 43 [0.90]	2.5	654 ± 35 [0.95]	3.2	652 ± 32 [0.95]	3.2
	10	10		2066 ± 119 [0.95]	1.8	2159 ± 58	1.8	2394 ± 52 [0.80]	1.6	1925 ± 44	2.0
f_{Rast}	2	50		524 ± 48 [0.95]	4.7	796 ± 68 [0.75]	3.1	569 ± 26 [0.35]	4.3	1365 ± 28 [0.10]	1.8
	5	140		4037 ± 209 [0.60]	2.6	5265 ± 313 [0.55]	2.0	13685 ± 257 [0.10]	0.8	7910 ± 82 [0.10]	1.3

between 1.4 and 3 for lmm-CMA. On these functions, nlmm-CMA is significantly more efficient. For the other tested functions f_{Rast}, f_{Schw} and $f_{\text{Schw}^{1/4}}$, nlmm-CMA is marginally more efficient than the standard lmm-CMA. In Fig. 2 and in Table 2 (highlighted rows), we evaluate the effect of increasing λ on nlmm-CMA using the same setting as in Section 3.2. Using population sizes larger than the default one, nlmm-CMA improves CMA-ES by a factor between 2.5 and 3.5 for all tested functions f_{Rosen}, $f_{\text{Schw}^{1/4}}$ and f_{Rast}. Therefore, nlmm-CMA maintains a significant speedup for λ larger than the default one contrary to lmm-CMA which offers a speedup approaching to 1 for f_{Rosen} and $f_{\text{Schw}^{1/4}}$ and a decreasing speedup (from 1.6 to 1.3) when λ increases (from 140 to 280) for f_{Rast}.

4.3 Impact of the Recombination Type

The choice of the recombination type has an important impact on the efficiency of ES in general [10] and CMA-ES in particular [2,3]. In the previous section, all the runs performed use the default weighted recombination type defined by Eq. 3. In the new variant of lmm-CMA, the meta-model acceptance criterion does not take into account the exact rank of each individual except the best one. By modifying the meta-model acceptance criteria of lmm-CMA, a possible accepted meta-model may be a meta-model that preserves the μ best individuals set and the best individual but generates a ranking far from the "true" ranking, i.e., the one based on the true objective function. We now compare nlmm-CMA using weighted recombination where weights are defined in Eq. 3 and intermediate recombination where weights are all equal to $1/\mu$: nlmm-CMA$_I$. Results are presented in Table 3. The algorithm nlmm-CMA outperforms nlmm-CMA$_I$ in all cases suggesting that even if the exact ranking is not taken into account

for assessing the quality of the meta-model in nlmm-CMA , this ranking is not random and has still an amount of information to guide CMA-ES.

4.4 Impact of Initial Parameters

In the tests presented so far, the initial parameters of the approximate ranking procedure (n_{init}, n_b) were initialized at the beginning of the optimization process to (λ, max[1, ($\frac{\lambda}{10}$)]). Every generation g, the number of initial individuals evaluated n_{init} is adapted (increased or decreased) depending on the meta-model quality (Lines 21 and 22 in Fig 3). The number of evaluations performed every generation is ($n_{init,g} + n_{ic,g} \times n_b$). We quantify now the impact of the initial values of (n_{init} and n_b) on the total cost of the optimization process. The algorithm nlmm-CMA is compared to a similar version where initial parameters are chosen as small as possible, i.e., n_{init} and n_b are equal to 1. Moreover, we consider two cases: (1) with update denoted nlmm-CMA$_1$, i.e., where initial parameters are adapted depending on the iteration cycle number (Lines 21 and 22 in Fig 3), and (2) without update denoted nlmm-CMA$_2$, i.e., parameters are equal to 1 during the entire optimization process (omitting lines 21 and 22 in Fig. 3). We note that in case (1), the number of evaluations for each generation g is ($n_{init,g} + n_{ic,g} \times n_b$) where $n_{init,0} = 1$ and $n_b = 1$. In case (2), every generation, lmm-CMA evaluates ($1 + n_{ic,g}$) individuals. The results on different test functions are summarized in Table 3.

On the unimodal functions f_{Schw}, $f_{\text{Schw}^{1/4}}$, setting n_{init} and n_b as small as possible in every generation, is marginally more efficient than the default definition of initial parameters on small dimensions except for dimension $n = 8$ and $\lambda = 10$. On f_{Rosen}, nlmm-CMA$_2$ is the most efficient compared to other approaches, except for dimension $n = 8$ and $\lambda = 10$ which can be justified by a higher number of unsuccessful runs compared to other approaches. On the multimodal function f_{Ack}, modifying the initial parameter n_{init} does not have an important impact on the speedup of lmm-CMA (between 1.5 and 4). However on f_{Rast}, using a small initial n_{init} decreases considerably the probability of success of the optimization, from 0.95 to between 0.35 and 0.10 for dimension $n = 2$ and $\lambda = 50$, and from 0.60 to 0.10 for dimension $n = 5$ and $\lambda = 140$. These results confirm the initial parameter choice suggested in [4].

5 Summary

In this work, we have investigated the performances of the lmm-CMA algorithm coupling CMA-ES with local meta-models. On f_{Rosen} and $f_{\text{Schw}^{1/4}}$, we have shown that the speedup of lmm-CMA with respect to CMA-ES drops to one when the population size λ increases. This phenomenon has been explained by the too restrictive condition used to stop evaluating new points dedicated at refining the meta-model, namely requiring that the *exact* ranking of the $\mu = \lambda/2$ best solutions is preserved when evaluating a new solution on the exact objective function. To tackle this problem, we have proposed to relax the condition to: *the*

set of μ best solutions is preserved and *the best individual* is preserved. The resulting new variant, nlmm-CMA outperforms lmm-CMA on the test functions investigated and the speedup with CMA-ES is between 1.5 and 7. Moreover, contrary to lmm-CMA it maintains a significant speedup, between 2.5 and 4, when increasing λ on f_{Rosen}, $f_{\text{Schw}^{1/4}}$ and f_{Rast}. The study of the impact of the recombination weights has shown that the default weights of CMA-ES are more appropriate than equal weights. The influence of two parameters, n_b and n_{init}, corresponding to the number of individuals evaluated respectively initially and in each iteration cycle has been investigated. We have seen that setting those parameters to 1 during the whole optimization process can marginally improve the performances on uni-modal functions and some multimodal test functions. However it increases the likelihood to be stuck in local minima for the Rastrigin function suggesting that the default parameter of lmm-CMA are still a good choice for nlmm-CMA.

Acknowledgments. The authors would like to thank Nikolaus Hansen for insightful discussions. This work was partially funded by the French National Research Agency (ANR) grant No. ANR-08-COSI-007.

References

1. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9(1), 3–12 (2005)
2. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
3. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. 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. 282–291. Springer, Heidelberg (2004)
4. Kern, S., Hansen, N., Koumoutsakos, P.: Local meta-models for optimization using evolution strategies. In: Parallel Problem Solving from Nature PPSN X, pp. 939–948. Springer, Heidelberg (2006)
5. 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)
6. Jebalia, M., Auger, A., Hansen, N.: Log linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. Algorithmica (accepted, 2010)
7. Auger, A., Hansen, N.: Performance evaluation of an advanced local search evolutionary algorithm. In: IEEE Congress on Evolutionary Computation, pp. 1777–1784 (2005)
8. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2009: Experimental setup. Research Report RR-6828, INRIA (2009)
9. Runarsson, T.P.: Approximate evolution strategy using stochastic ranking. In: IEEE Congress on Evolutionary Computation, pp. 745–752 (2006)
10. Arnold, D.V.: Optimal weighted recombination. In: Wright, A.H., Vose, M.D., De Jong, K.A., Schmitt, L.M. (eds.) FOGA 2005. LNCS, vol. 3469, pp. 215–237. Springer, Heidelberg (2005)

Exploiting Evolution for an Adaptive Drift-Robust Classifier in Chemical Sensing

Stefano Di Carlo[1], Matteo Falasconi[2], Ernesto Sánchez[1], Alberto Scionti[1], Giovanni Squillero[1], and Alberto Tonda[1]

[1] Politecnico di Torino – Torino, Italy
[2] Università di Brescia & SENSOR CNR-INFM– Brescia, Italy

Abstract. Gas chemical sensors are strongly affected by drift, i.e., changes in sensors' response with time, that may turn statistical models commonly used for classification completely useless after a period of time. This paper presents a new classifier that embeds an adaptive stage able to reduce drift effects. The proposed system exploits a state-of-the-art evolutionary strategy to iteratively tweak the coefficients of a linear transformation able to transparently transform raw measures in order to mitigate the negative effects of the drift. The system operates continuously. The optimal correction strategy is learnt without a-priori models or other hypothesis on the behavior of physical-chemical sensors. Experimental results demonstrate the efficacy of the approach on a real problem.

Keywords: real-valued function optimization, parameter optimization, real-world application, chemical sensors, and artificial olfaction.

1 Introduction

Chemical sensing is an intrinsically challenging task. Artificial olfaction [1], also known as *electronic nose*, tries to mimic human olfaction by using arrays of gas chemical sensors together with pattern recognition (PARC) techniques. The most common class of sensors used for chemical sensing is metal oxide semiconductors (MOX) [2]. When sensors come in contact with volatile compounds, the adsorption of these elements on the sensor's surface causes a physical change of the sensor itself, and hence, a change of its electrical properties. Each sensor is potentially sensitive to all volatile molecules in a specific way. The response of the sensor is recorded by its electronic interface, and the corresponding electrical signal is then converted into a digital value. Recorded data are finally elaborated and classified by PARC algorithms (a brief overview of which is presented in Section 2.1) often based on statistical models [3]. The power and appropriateness to data of the PARC strategy determine the final performance of the electronic nose. Yet, independently on that, chemical sensors drift can ultimately invalidate some of the classification models.

Sensor drift is defined as the temporal shift of sensors' response under constant environmental (physical and chemical) conditions. Sensor drift is one of the most serious impairments afflicting all kinds of sensors such as for instance pressure sensors [4], pH sensors [5], conductivity sensors [6], as well as chemical sensors [7]. Sensor drift originates from unknown dynamic processes in the sensor device typically related to sensing

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 412–421, 2010.

material modifications. These modifications are usually caused by irreversible phenomena such as poisoning or aging. Nowadays, the only effective counteraction to prevent negative effects of drift is frequent sensor calibration. However, while this approach is rather simple to implement for physical sensors where the quantity to be measured is exactly known, chemical sensors pose a series of challenging problems. Indeed, in chemical sensing, the choice of the calibrant strongly depends on the specific application and, when the sensing device is composed of a number of cross-correlated sensors, a univariate signal correction is not feasible.

Long-term drift produces dispersion in the patterns that may change the clusters distribution in the data space. As a consequence, learnt classification boundaries may turn completely useless after a given period of time. Methods for algorithmically correcting, retraining or physically minimizing sensor drift are therefore highly requested by any robust chemical sensing system.

Drift correction algorithms are not new in the field [1] (chapter 13). Notwithstanding the first attempts to tackle this problem date back to early 90s, the study of drift is still a challenging task for the chemical sensor community (see Section 2.2).

In this paper we propose a novel architecture for an evolutionary adaptive drift-robust classifier (Section 3). A linear transformation is applied to the raw measures read by the electronic nose. Such linear transformation is initially described by the identity matrix, and slowly evolved to compensate drift effects in the classification process. The evolutionary core is continuously active, monitoring the performance of the classification and adjusting the matrix coefficients on each new measure. It makes use of a covariance matrix adaptation evolution strategy (CMA-ES), perfectly suited for solving difficult optimization problems in continuous domain. The output of the classifier is used to calculate the fitness function. Compared to existing adaptive solutions, the proposed approach is able to transparently adapt to changes in the sensors' responses even when the number of available samples is not high and new classes of elements are introduced in the classification process at different time frames. To prove this we tested our approach on an experimental data set for detection and classification of chemical compounds by a gas sensor array (Section 4).

2 Background

2.1 Pattern Classification

Pattern recognition is a widely addressed topic in the literature [8]. Automatic recognition, description, classification, and grouping of patterns are important problems in a variety of engineering and scientific disciplines such as biology, psychology, medicine, marketing, computer vision, artificial intelligence, remote sensing, and artificial olfaction as well [3].

The primary goal of pattern recognition is supervised or unsupervised classification. In supervised classification the dataset is normally composed of two sub-sets of samples: an initial set called *training set* made of known samples used to train the classifier, and a second group called *test set* composed of unknown samples to classify.

After the signal pre-processing stage, the first critical step in the classification process is feature extraction and/or selection. This phase involves several steps designed to organize and reduce the data space dimensionality, and to avoid problems associated with high-dimensional sparse datasets (course of dimensionality). During the training phase, feature extraction/selection algorithms find the appropriate features to represent the input patterns, and the classifier is trained to partition this resulting feature space.

Dimensionality reduction techniques are also employed for data visualization in order to have a preliminary insight of the multidimensional pattern distribution. Techniques for visualizing and projecting multidimensional data, along with cluster analysis methods, are also referred to as *exploratory data analysis*. The most used exploratory analysis method is principal component analysis (PCA). PCA takes linear combinations of initial variables to identify the directions of maximum variance of the data (called principal components). Typically, only the first two or three components - exploring the highest variance - are retained for visualization purposes, but this is generally enough to understand how data are clustered, i.e., the position and shape of clusters.

Among the various frameworks in which pattern recognition has been traditionally formulated, the statistical approach has been most intensively studied and used in practice [9]. Under this framework several classification algorithms have been proposed and extensively used for chemical sensing, such as: linear discriminant analysis (LDA), k–nearest neighbors (KNN) [10], and more recently support vector machines (SVM) [11] and random forests [12]. Neural networks (NNET) [13] and methods imported from statistical learning theory also received special attention in the field of artificial olfaction [1] (chapter 6).

2.2 Drift Compensation Approaches for Chemical Sensing

Several methods have been proposed to tackle the problem of drift compensation for chemical sensing [1] (chapter 13). Current solutions fall into three categories: (a) use of calibrants to return the classifier to its original state; (b) attune the classifier with proper feature selection/extraction to reduce drift effects; (c) use of “adaptive” models to real-time update the classifier.

Use of a single calibrant or a set of calibrants to retrain a classifier is perhaps the only robust method for determining precise information regarding the degradation of the classification model regardless of the degree of sensor drift [14]. It is also the only method able to sustain a high degree of classification performance even in presence of inconsistent sensor drift over an extremely long period of time. Nevertheless, calibration is the most time-intensive method for drift correction since it requires system retraining. Hence, it should be used sparingly. Moreover, the calibrant selection must be accurately chosen depending on the application. This leads to loss of generalization and lack of standardization, which would be highly required by industrial systems.

Attempts to attune the classifier to specific features of interest have been used in conjunction with both PCA [15] and independent components analysis (ICA) [16] to determine which dimensions of the analyte space most highly correlate with the differences between the analytes in the set. These presumably represent the dimensions that are least noisy and/or are least affected by drift and therefore are the only ones retained in constructing the classification model. Attuning methods can

provide significant improvements in classification over a fixed time period. However, adding new analytes to the recognition library represents a major problem since the previously rejected dimensions might be necessary to robustly identify these new classes. Additionally, these methods contain no provisions for updating the model, and thus may ultimately be invalidated by time evolving drift effects.

Adaptive models try to on-line adjust the classifier by taking into account pattern changes due to drift effects. Neural networks, such as self-organizing maps (SOMs) [17] or adaptive resonance theory (ART) networks [18], have been frequently used in the past. Under such schemes, newly recognized data that match the stored analyte fingerprints, i.e., processed measures used as input for the classifier, can be continuously used to retrain the classifier. This technique has the advantage of simplicity because no recalibration is required. Yet, two main weaknesses can be identified. First, a discontinuity in response between two consecutive exposures (regardless of the time interval between the exposures) would immediately invalidate the classification model and would prevent adaptation. Second, a key to obtain reliable results is to set appropriate thresholds for choosing the winning neuron, and this typically requires a high number of training samples owing to the complexity of the network topology.

3 Proposed Architecture

Figure 1 graphically represents the architecture exploited in this paper to implement a drift-robust classification system for chemical sensing. The proposed approach couples a standard classifier with an adaptive mechanism able to compensate drift effects. It is important to highlight that the classifier itself is not modified, and no additional training is required.

The basic idea is to map the vector of raw measures associated to each sample analyzed by the electronic nose (Rm) with a new fingerprint used as input for the classifier (Fp). Fp is obtained by applying a linear transformation represented by a square matrix C to Rm ($Fp = C \times Rm$).

The classifier receives the produced fingerprints and provides as output a partition of the test set in a given number of classes. The performance of the full classification system can be measured in terms of the percentage of fingerprints correctly classified.

The matrix C evolves while the system is operating with the goal of supporting the classifier in compensating the drift effects. At the beginning C is initialized to the identity matrix, hence, no modification is performed. Whenever a new sample is collected and classified, an evolutionary optimizer slightly tweaks the matrix coefficients in order to increase the robustness of the classification.

The evolutionary scheme is backed up by an *adaptation manager* in charge of deciding whether to activate the evolutionary core that updates the matrix C depending on the classification confidence delivered by the classifier for every fingerprint.

The resulting system is potentially able to compensate any possible drift with the reasonable assumption that the drift should be a relatively slow phenomenon compared to the sampling rate. This allows describing the disturbance as a continuous function over a limited number of consecutive measures. Since parameters are adapted seamlessly, the system is able to compensate disturbance up to quite relevant magnitudes. Considering the drift as a slow phenomenon implies that the initial training data can be considered not affected by it.

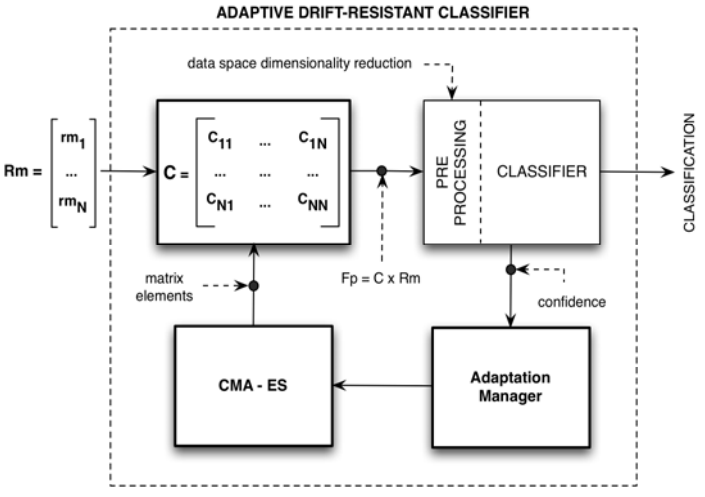


Fig. 1. Main architecture of the adaptive drift-resistant classifier

As evolutionary optimizer we exploit a covariance matrix adaptation evolution strategy. Briefly speaking, an evolution strategy (ES) is a stochastic, population-based, iterative optimization method belonging to the class of evolutionary algorithms, devised in the early 60s by Ingo Rechenberg and Hans-Paul Schwefel. An ES represents an individual as a vector of real-valued numbers. Mutation is performed by adding a normally distributed random value to each vector component. Generating the offspring through mutation corresponds to a sampling in the solution space. The ES is able to determine the optimal value of some of its parameters. Remarkably, the *step size*, i.e., the standard deviation of the normal distribution used for sampling the mutation, is usually self-adapted. The *covariance matrix adaptation* (CMA) is a method to update the covariance matrix of the multivariate normal mutation distribution in the ES. New candidate solutions are generated according to the mutation distribution. Original ES implementations simulate the evolution at the level of species and do not include any recombination operators, although later implementations often do.

The covariance matrix describes the pair-wise relationships between the variables in the distribution. CMA-ES represents the latest breakthrough in the ES field [19]. Results reported in the literature demonstrate that it can easily tackle problems where the fitness landscape presents discontinuities, sharp bends or ridges, noise, and local optima.

In the presented approach, each time a fingerprint is classified with a confidence in a range delimited by an upper-bound and a lower-bound threshold, the adaptation manager enables the CMA-ES. The probability estimates produced by the classifier for the predicted class can be exploited as a measure of the confidence of the classifier in its prediction. This confidence is the fitness value of an individual for the CMA-ES. The two thresholds allow identifying which fingerprints to use to evolve the system, i.e., update the linear transformation *C*, in order to adapt to the drift modifications. The lower threshold aims at discarding spurious fingerprints, i.e., fingerprints whose classification confidence is too low and therefore do not provide reliable

information for the given class, making the proposed framework resistant to spikes. The upper threshold identifies fingerprints that are already corrected in a satisfactory way by the current linear transformation and could not be further enhanced in a sensible way. Skipping these samples allows to reduce the computation effort by activating the evolution only for samples where the effect of the drift becomes more visible.

In order to maximize the correction ability of the proposed architecture, the linear transformation is computed considering the selected fingerprint and a group of K previous fingerprints. The current matrix C is applied to the current measure and to the K previous raw samples. Thus, the evolution continues until the classification confidence for the current fingerprint reaches the upper threshold, and the classification confidence of the K previous ones is not decreased more than ε percent when using the current matrix C.

The final linear transformation matrix is obtained as a linear combination between the current linear transformation matrix and the new one obtained by the application of the CMA-ES, as follows: $C_n = C_n \cdot \alpha + C_{n-1} \cdot (1-\alpha)$, where C_n and C_{n-1} are respectively the new and the current linear transformation matrices, and α is a parameter modeling the *inertia* of the system when moving from the old to the new transformation. This limits the possibility of generating linear transformation matrices unable to improve the classification process. Both ε and α are input parameters of the proposed system with values ranging in the interval $[0,1]$.

4 Experimental Results and Discussion

The proposed approach has been validated on a real data set collected at *SENSOR Lab*, an Italian research laboratory specialized in the development of chemical sensor arrays (more information is available at *http://sensor.ing.unibs.it/*).

The data under consideration have been obtained using the EOS835 *electronic nose* composed of 6 chemical MOX sensors (further information on sensors and equipment can be found in the review paper [2] and its references). The main goal of the performed experiments is to determine the capability of the EOS835 to identify five pure organic vapors, namely: ethanol (1), water (2), acetaldehyde (3), acetone (4), ethyl acetate (5). These are common chemical compounds to be detected in real-world applicative scenarios such as food industry, industrial processes, and biomedical field.

Five different measurement sessions were performed during a period of time of about one month. The elapsed time, though not very long, is enough to obtain data affected by a certain amount of drift.

Different classes have been introduced during the different measurement sessions, a common practice in real world experiments: classes 1 and 2 are measured since the beginning, class 3 is first introduced during the second session (one week later), while classes 4 and 5 appear only during the third session (ten days after the beginning of the experiment).

The number of samples contained in the data set is high (545 measurements) if compared to datasets reported in the literature. It must be noticed that performing measurements with arrays of chemical sensors is a time consuming task that limits the amount of samples that can be produced.

There is not a perfect balance among the number of measurements belonging to every class, with a clear predominance of classes 1, 2 and 3 over classes 4 and 5. This perfectly respects real situations, and additionally increases the difficulty of properly compensating the sensor drift.

PCA plot in figure 2 clearly shows the presence of sensor drift in the considered data set, i.e. shift of samples in a direction over the time. Measures of the test set tend to drift toward a direction that is perfectly visible on the first two principal components. This phenomenon leads to an overlapping of the different classes and, consequently, to a loss of classification performance as time goes on.

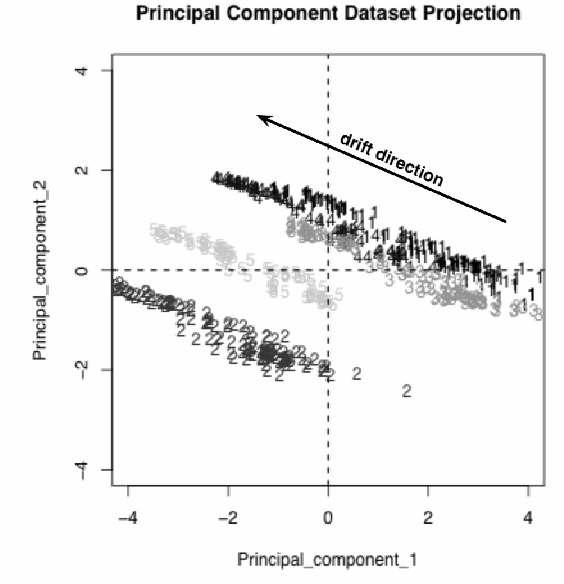


Fig. 2. PCA plot for the test set. The effect of the drift is visible as a shift of the samples over the time towards the upper-left corner.

The classifier used in the experience is a *linear discriminant analysis* (LDA) classification algorithm implemented using the R scripting language [20], while the adaptation block is implemented in C and PERL.

The training set for the classifier is composed of the first 20 measurements of each class, while the remaining 445 samples have been used as test set. The considered training set is quite small compared to common experimental setups that tend to use about one third of the samples to build the prediction model. Nevertheless, this allows to work with a reasonable number of test samples that can be used to better test the evolutionary approach. In order to better understand the characteristics of the considered data model, a preliminary classification of the test set performed with the LDA classifier without drift compensation has been performed. The classifier by itself performs relatively well, with about 90% of the test set correctly classified. Only 43 samples are incorrectly classified. Figure 3-A summarizes this result with a confusion matrix, i.e., a matrix that shows how many samples belonging to a class A have been correctly assigned to class A or misclassified w.r.t. true labels. The figure highlights

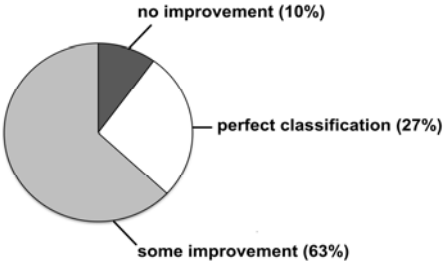
how critical samples belonging to class 1, due to the drift, tend to overlap with classes 3 and 4.

The proposed correction system has been applied considering the following parameters, set after 100 experiments aimed at tuning the framework: $\alpha = 0.99$, $\varepsilon = 1$, the lower and upper bounds respectively set to 0.65 and 0.99, and $K = 40$. Small differences in the adopted parameters haven't led to better performance for the proposed system.

The CMA-ES does not require parameter tuning for its application: finding good strategy parameters is considered as part of the algorithm design. In fact, the choice of strategy internal parameters is not left to the user, with the exception of the population size $\lambda = 100$ and the initial standard deviation set to 10^{-2} because of the order of magnitude of the measures in the experiments.

PREDICTION \ REFERENCE	1	2	3	4	5
1	70	0	0	0	0
2	0	110	0	0	0
3	2	0	92	0	0
4	41	0	0	66	0
5	0	0	0	0	64

(A)



(B)

Fig. 3. (A) Confusion matrix for the not corrected LDA classifier and (B) overall results for the robust classifier

In order to experimentally validate our approach, 150 repetitions of the classification experiments were executed using the previously presented setup (figure 3-B). The proposed approach has improved the original performance of the classifier in 90% of the runs while none of the executions worsened the initial classification. In 27% of the runs all 445 raw measures belonging to the test set have been correctly classified. 63% of the runs produced classification results with an average performance of about 97% approaching the theoretical performance of 100% computed for the LDA algorithm performing cross validation on the training set with 20 iterations and 5% of samples left out at each iteration. The proposed approach can be used for a real-time measurement correction since the time required to obtain a good adjustment (up to 20 mins) is much smaller compared with drift time scale. Moreover it is comparable with the sensors' recovery time.

5 Conclusions

This paper presented an adaptive drift-robust classification system based on an evolutionary strategy. To date, this is the first attempt to exploit evolutionary algorithms capabilities to approach the drift problem in chemical sensing.

The obtained results on a real dataset are encouraging. The classification performances increase to a level close to the theoretical upper bound for the chosen classification algorithm. Yet, while the results are encouraging in terms of improvement of the classification rate, the main drawback of the proposed approach is that it is not able to produce a clear representation of the drift and actually remove it from the row measures. The parameters found by the CMA-ES slightly shift the raw measures in a direction usually orthogonal to the drift, obtaining a better separation of the different classes that allows the LDA classifier to discriminate more accurately among them. This is mainly due to the fact that the evolutionary core of our system does not include any model of the drift that could be exploited to drive the evolution toward a real elimination of the drift effect on the row measures. We are currently working towards the inclusion of several heuristics and considerations on spatial measures in the fitness values, to allow the CMA-ES to both increase the classification rate and perform drift correction. This includes considering different models for the drift effect on chemical sensors.

Comparison between artificially data sets and real data sets will be used to validate our framework.

Acknowledgements

We would like to thank Isabella Concina for technical assistance during experimental measurements.

References

[1] Pearce, T.C., Shiffman, S.S., Nagle, H.T., Gardner, J.W.: Handbook of machine olfaction. Wiley-VHC Ed., Weinheim (2003)
[2] Pardo, M., Sberveglieri, G.: Electronic olfactory systems based on metal oxide semiconductor sensor arrays. MRS Bulletin 29(10), 703–708 (2004)
[3] Gutierrez-Osuna, R.: Pattern Analysis for Machine Olfaction: A Review, June 2002, vol. 2, pp. 189–202 (2002)
[4] Polster, A., Fabian, M., Villinger, H.: Effective resolution and drift of Paroscientific pressure sensors derived from long-term seafloor measurements. Geochem. Geophys. Geosyst. 10(Q08008) (2009)
[5] Chen, D.Y., Chan, P.K.: An Intelligent ISFET Sensory System With Temperature and Drift Compensation for Long-Term Monitoring. IEEE Sensor Journal 8(11-12), 1948–1959 (2008)
[6] Owens, W.B., Wong, A.P.S.: An improved calibration method for the drift of the conductivity sensor on autonomous CTD profiling floats by theta–S climatology. Deep-Sea Research Part I-Oceanographic Research Papers 56(3), 450–457 (2009)
[7] Aliwell, S.R., et al.: Ozone sensors based on WO3: a model for sensor drift and a measurement correction method. Measurement Science & Technology 12(6), 684–690 (2001)
[8] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, Hoboken (2000)
[9] Jain, A.K., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 4–37 (2000)

[10] Dasarathy, B.V. (ed.): Nearest neighbor (NN) norms: Nn pattern classification
[11] Pardo, M., Sberveglieri, G.: Classification of electronic nose data with support vector machines. Sensors and Actuators B: Chemical, 730–737, June 29 (2005)
[12] Pardo, M., Sberveglieri, G.: Random forests and nearest shrunken centroids for the classification of sensor array data. Sensors And Actuators B-Chemical 131, 93–99 (2008)
[13] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford Univ. Press, Oxford (1995)
[14] Sisk, B.C., Lewis, N.S.: Comparison of analytical methods and calibration methods for correction of detector response drift in arrays of carbon black-polymer composite vapor detector. Sensors and Actuators B: Chemical, 249–268, January 24 (2005)
[15] Artursson, T., et al.: Drift correction for gas sensors using multivariate methods. Journal of Chemometrics 14, 711–723 (1999); Special Issue: Proceedings of the SSC6, HiT/TF, Norway (August 1999)
[16] Di Natale, C., Martinelli, E., D'Amico, A.: Counteraction of environmental disturbances of electronic nose data by independent component analysis. Sensors and actuators. B, Chemical 82(2-3), 158–165 (2002)
[17] Marco, S., Ortega, A., Pardo, A., Samitier, J.: Gas Identification with Tin Oxide Sensor Array and Self-Organizing Maps: Adaptive Correction of Sensor Drifts. IEEE Transactions on Instrumentation and Measurement 47, 316–321 (1998)
[18] Vlachos, D.S., Fragoulis, D.K., Avaritsiotis, J.N.: An adaptive neural network topology for degradation compensation of thin film tin oxide gas sensors. Sensors and Actuators B: Chemical, 223–228, December 15 (1997)
[19] Hansen, N.: The CMA evolution strategy: a comparing review, towards a new evolutionary computation. In: Lozano, J.A., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) Advances on estimation of distribution algorithms, pp. 75–102. Springer, Heidelberg (2006)
[20] Kuhn, K.: Building Predictive Models in R Using the caret Package. Journal of Statistical Software 28(5), 1–26 (2008)

Automatically Modeling Hybrid Evolutionary Algorithms from Past Executions

Santiago Muelas, José-María Peña, and Antonio LaTorre

DATSI, Facultad de Informática
Universidad Politécnica de Madrid, Spain
{smuelas,jmpena,atorre}@fi.upm.es

Abstract. The selection of the most appropriate Evolutionary Algorithm for a given optimization problem is a difficult task. Hybrid Evolutionary Algorithms are a promising alternative to deal with this problem. By means of the combination of different heuristic optimization approaches, it is possible to profit from the benefits of the best approach, avoiding the limitations of the others. Nowadays, there is an active research in the design of dynamic or adaptive hybrid algorithms. However, little research has been done in the automatic learning of the best hybridization strategy. This paper proposes a mechanism to learn a strategy based on the analysis of the results from past executions. The proposed algorithm has been evaluated on a well-known benchmark on continuous optimization. The obtained results suggest that the proposed approach is able to learn very promising hybridization strategies.

1 Introduction

The selection of the most appropriate Evolutionary Algorithm (EA) for a given optimization problem is a difficult task, sometimes considered an optimization problem itself [2].

Even though the No Free Lunch Theorem asserts that "*any two algorithms are equivalent when their performance is averaged across all possible problems*", in practice, and being constrained to certain types of problems, the performance of some particular algorithms is better than others. In most of the cases, the selection of the most appropriate algorithm is carried out by the execution of several alternative algorithms (advised by the literature or the own experience) and then choosing the one reporting the best results.

Supported by these arguments, hybrid evolutionary techniques are a promising alternative to deal with these situations. By combining different heuristic optimization approaches, it is possible to profit from the benefits of the best approach, avoiding the limitations of the others. These hybrid algorithms also hold the hypothesis that the combination of several techniques can outperform the sole usage of its composing algorithms. This hypothesis is based on the idea that the comparative performance of the algorithms is not the same along the whole optimization process. Moreover, it is possible to identify different best performing algorithms for different phases of the optimization process.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 422–431, 2010.

Nowadays, there is an active research in the design of dynamic or adaptive hybrid algorithms. However, this paper introduces a different perspective, barely explored in the literature. This contribution proposes a mechanism to learn the best hybrid strategy from the analysis of the results from past executions. The idea of using past executions to induce the most appropriate hybridization technique is particularly useful in those scenarios in which an optimization problem is solved multiple times. These multiple executions could include slightly different conditions that actually have an influence in the position of the optimal value, but do not change the main characteristics of the fitness landscape in which this optimization process searches. Many industrial problems have this characteristic in which the fitness function is mainly the same, but the particular conditions or other input parameters are changed on each execution, e.g., the optimization of engineering structures evaluated under different stress conditions.

A rather naïve solution to this approach is the design of an alternating strategy, based on the generation number or the fitness values. Nevertheless, this idea does not consider that reaching a given fitness value or a particular generation number is achieved via a stochastic process. This process does not ensure that the same generations or the same fitness values reached by an algorithm actually represent the same situation of the search process in two different executions. Any successful strategy would need not only this general parameters, but also other statistical or introspective information of the evolving population, in order to identify a situation similar to one previously learned.

This paper presents a new hybrid Evolutionary Algorithm that learns the best sequence of techniques according not only to their performance but also to other statistical/informative parameters of the evolved population. This new algorithm has been evaluated using a well-known benchmark of continuous optimization functions and its results have been validated using non-parametric tests.

The rest of the paper is organized as follows: Section 2 presents an overview of several hybrid algorithms. Section 3 details the proposed algorithm. In Section 4 the experimental scenario is described in detail. Section 5 presents and comments on the results obtained and lists the most relevant facts from this analysis. Finally, Section 6 contains the concluding remarks obtained from this work.

2 Related Work

In this section, some of the most relevant work on High-level relay hybrid (HRH) algorithms will be reviewed. The HRH terminology was introduced in [7], one of the first attempts to define a complete taxonomy of hybrid metaheuristics. This taxonomy is a combination of a hierarchical and a flat classification structured into two levels. The first level defines a hierarchical classification in order to reduce the total number of classes, whereas the second level proposes a flat classification, in which the classes that define an algorithm may be chosen in an arbitrary order. From this taxonomy, four basic hybridization strategies can be derived: (a) LRH (Low-level relay hybrid): One metaheuristic is embedded into a single-solution metaheuristic. (b) HRH (High-level relay hybrid): Two

metaheuristics are executed in sequence. (c) LTH (Low-level teamwork hybrid): One metaheuristic is embedded into a population-based metaheuristic. (d) HTH (High-level teamwork hybrid): Two metaheuristics are executed in parallel. For this work, we have focused on the HRH group, the one the algorithm proposed in this paper belongs to.

There has been an intense research in HRH models in the last years combining different types of metaheuristics. In the following paragraphs some of the most recent and representative approaches will be reviewed.

The DE algorithm is one of the evolutionary algorithms that has been recently hybridized following the HRH strategy. For example, it has been combined with Evolutionary Programming (EP) in [9]. The EP algorithm is executed for each trial vector created by the DE algorithm which is worse than its associated target vector. DE has also been combined with PSO [3]. In this case, the PSO algorithm is executed as the main algorithm but, from time to time, the DE algorithm is launched to move particles from already explored areas to new positions. The particles preserve their velocity when they are moved by the DE in order to minimize the perturbation in the general behavior of the PSO algorithm.

There have also been some studies that have tried to use adaptive learning for combining the algorithms. In [6], the authors propose two adaptive strategies, one heuristic and one stochastic, to adapt the participation of several local searches when combined with a Genetic Agorithm (GA). In both strategies, there is a learning phase in which the performance of each local search is stored and used in later generations in order to select the local search to apply. In [1,10] several local searches are combined with a metaheuristic algorithm using also an adaptive scheme. The application of each algorithm is based on a population diversity measure which varies among the studies. When applied to the DE algorithm, this strategy prevents the stagnation problems of the DE by reducing the excessive difference of the best individual and the rest of the population.

Finally, as far as the authors are concerned, no other study has ever tried to focus on doing a post-execution learning of the best patterns for combining the algorithms of a HRH algorithm. Therefore, this contribution proposes a new approach, based on this idea, to try to exploit the potential synergies between different search strategies.

3 Contribution

In this study, we propose the hypothesis that it is possible, based on the behavior of previous executions, to learn a hybridization strategy for the algorithms of an HRH algorithm in order to select the most appropiate algorithm for each iteration of the execution.

For this task, a new methodology for executing HRH algorithms has been developed. Briefly, the methodology "observes" the execution of a HRH algorithm and stores some information about each state of the execution along with the information (for that state) of the performance of the algorithms involved in the hybrid algorithm. With this information, the methodology is able to construct a

model and use it as a hybridization strategy of a new algorithm which will try in future executions to select the most appropiate algorithm for each state found.

The main steps of this proposal are depicted in Figures 1.a and 1.b. As previously mentioned, the methodology starts the execution of a HRH algorithm which, as all hybrid algorithms, has a hybridization strategy that determines the combination of the algorithms involved in the hybrid. Let P_i be the population of iteration i and the *active algorithm* the one selected by the hybridization strategy for executing at that iteration. First, the *active algorithm* is executed over P_i for M evaluations (*period of evaluations*) generating the population P_{i+M}^{active}. In order to compare its performance, the remaining algorithms are also executed with the same starting population P_i, generating a new population P_{i+M}^{j} for each j algorithm. The algorithm that produces the individual with the highest score from P_{i+M}^{active} and all P_{i+M}^{j} populations is granted a *win*. Since the involved algorithms are stochastic, this process needs to be repeated several times in order to obtain a more reliable measure of the performance. After N repetitions starting with the same population P_i, the methodology generates a data record which stores the information of the state and the number of wins of each of the involved algorithms. The state of an iteration is determined by some extracted measures from both the population and the hybridization strategy. The execution continues with the population of the *active algorithm* of the last repetition and continues this process until the stop criterion is satisfied.

After a number of executions, a broad data set of data records, which store the performance of the algorithms over different states, is obtained. Then, the records are preprocessed, filtering out those which have the same number of wins for all the algorithms and adding a class attribute which contains the algorithm with the highest number of wins. The resultant data set is used as input for a machine learning algorithm (c4.5 in this study), which returns a set of rules that determine the best algorithm to apply at each state. This model is used to construct the proposed smartHRH algorithm which, at each iteration, analyzes the state and selects (according to the model) the most appropriate algorithm.

Since the proposed methodology can be applied to any HRH algorithm, the described process can also be successively applied to the smartHRH algorithm to refine the learned model. To infer the model of each version of smartHRH, the previous data sets of data records are also considered. Each data set participates with the same number of records, therefore, the maximum number which are sampled is determined by the smallest data set.

Although the proposed process can be started with any HRH algorithm, an HRH algorithm which tries to select the best combination of algorithms has been used as the initial algorithm. This algorithm, called baseHRH, uses the information of the number of wins described earlier in order to select the best algorithm for a specific state. It follows the same steps of Figure 1.a but continues the execution with the population of the last attempt of the algorithm that obtained the greatest number of wins. This way, the initial set of records used for learning come from a potentially good sequence of algorithms.

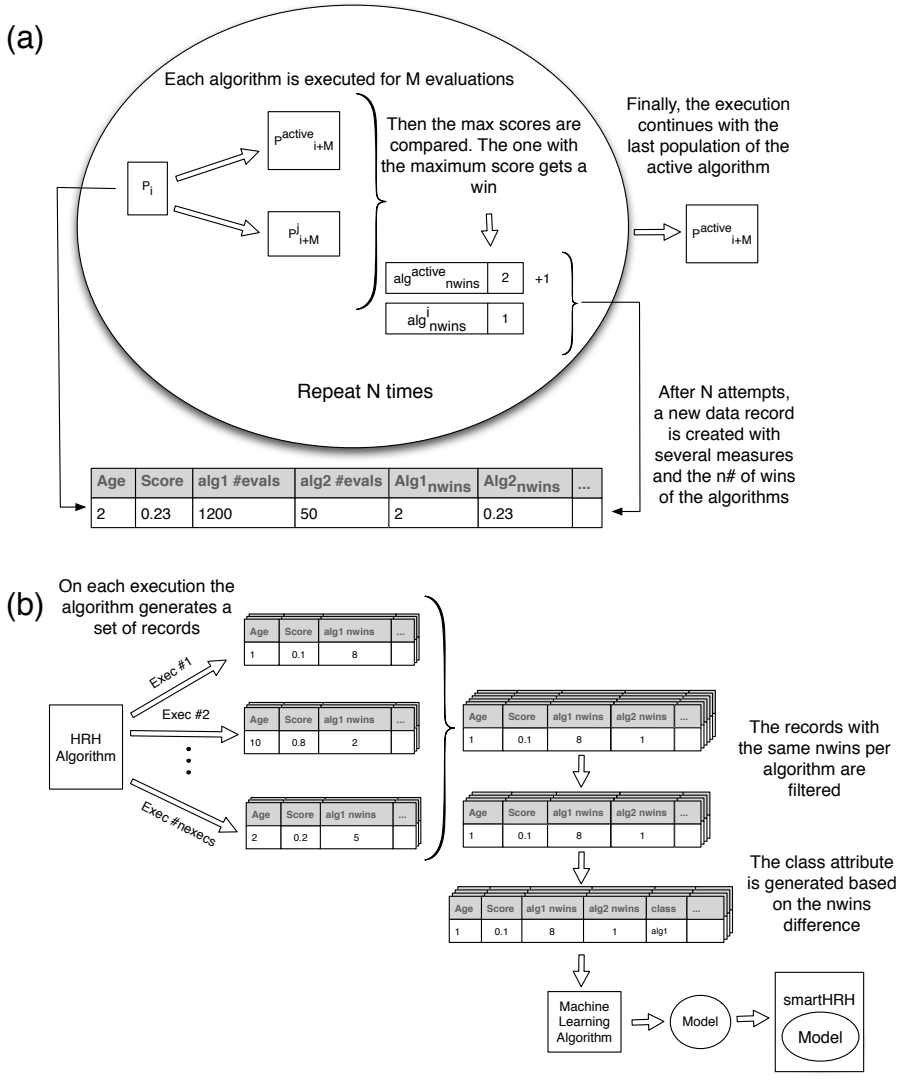


Fig. 1. Generation of the data records and learning procedure

4 Experimentation

For this experimentation, the benchmark from the workshop on *Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test* held at the ISDA 2009 Conference has been considered. This benchmark defines 11 continuous optimization functions. The first 6 functions were originally proposed for the '*Special Session and Competition on Large Scale Global Optimization*" held at the CEC 2008 Congress [8]. The other 5 functions

Table 1. Functions

Id	Name
f1	Shifted Sphere Function
f2	Shifted Schwefel's Problem 2.21
f3	Shifted Rosenbrock's Function
f4	Shifted Rastrigin's Function
f5	Shifted Griewank's Function
f6	Shifted Ackley's Function
f7	Schwefel's Problem 2.22
f8	Schwefel's Problem 1.2
f9	Extended f_{10}
f10	Bohachevsky
f11	Schaffer

Table 2. DE Parameter Values

Parameter	Values
Population size	25
CR	0.5
F	0.5
Crossover Op.	Exponential
Selection Op.	Tournament 2

have been specially proposed for the Workshop of the ISDA 2009 Conference. These functions, presented in Table 1, have different degrees of difficulty and can scale to any dimension. Detailed information about the selected benchmark can be found at the web page of the organizers of the workshop[1].

The results reported in this section are the aggregation of 25 independent executions on 200 dimensional functions. The performance criterion is the distance (error) between the best individual found and the global optimum in terms of fitness value. Following the benchmark recommendations, the maximum number of Fitness Evaluations has been fixed to $5000 \times D$, where D is the number of dimensions. Due to the constraints of the framework employed, the maximum reachable error without loosing precision is 1E−14.

The algorithms that were used for the new HRH algorithm are two which combination obtained very competitive results on the workshop of the ISDA 2009 Conference [5]. These two are the DE algorithm and the first of the local searches of the MTS algorithm [11]. The MTS algorithm was designed for multi-objetive problems but it has also obtained very good results with large scale optimization problems. In fact, it was the best algorithm of the CEC'08 competition [8]. The DE algorithm is one of the recent algorithms that, due to its results, has quickly gained popularity on continuous optimization. In the last IEEE competitions on continuous optimization, a DE-based algorithm has always reached one of the best three positions. Nevertheless, DE is subject to stagnation problems which could heavily influence the convergence speed an the robustness of the algorithm [4]. Therefore, the idea of combining them is to assist the explorative power of DE by an exploitative local search which has proven to obtain some of the best results. The reason for selecting only the first of the three local searches of the MTS is that, in a previous study by the authors on the same set of functions, this local search was the one that achieved the best results. Besides, we have slightly modified this local search so that, at each iteration, it only explores a subset of randomly

[1] http://sci2s.ugr.es/programacion/workshop/Scalability.html

selected dimensions (75% of the total). This modification has achieved a similar performance with a 25% less of evaluations on a preliminary analysis allowing the hybrid algorithm to spend more evaluations in the DE algorithm. The parameters used for the DE are the same ones of a hybrid algorithm presented at the ISDA 2009 Conference [5] and are presented in Table 2. Any measure could be used for specifying a state but, for the experiments, the following were selected: maximum age, number of evaluations, total and average number of evaluations per algorithm, number of activations of each algorithm and the ration of one against the other, number of evaluations of the best individual without improvement and the best score.

Finally, the period of evaluations used to compare the performance of both algorithms has been set to 1250 evaluations, a value that obtained good results in a set of previous tests.

For analyzing the results, the following algorithms were executed over the benchmark: the DE algorithm, the first local search of the MTS algorithms (LS1), a randomHRH algorithm which, at each iteration, selects an algorithm based on a binomial distribution of $p = 1/2$, the baseHRH algorithm used for obtaining the first model described in Section 3 and the best smartHRH algorithm. Up to eight versions of smartHRH algorithms were obtained per function. From those eight, the one with the best average score was selected. Then, the algorithm was executed again using the same model of this best execution in order to be fair with the remaining algorithms.

5 Analysis of the Results

Table 3 presents the results of the average score of the 25 executions. The best values for each function are highlighted in the table. It can be seen that the smartHRH algorithm obtains the best results in 9 out of 11 functions, reaching the global optimum[2] in 8 functions. It can also be seen that the smartHRH algorithm outperforms the other HRH algorithms (random and base) in most of the functions. In order to obtain a better comparison, each algorithm was compared against each other using the non-parametric Wilcoxon signed-rank test. Each cell $D_{i,j}$ in Table 4 displays the functions for which the results of algorithm i were signicantly better than those of algorithm j with a $p - value < 0.05$. Here, the smartHRH algorithm is also the clear winner obtaining better results than any other algorithm in at least four functions whereas it only looses against randomHRH in f2 and against DE and randomHRH in f3. In these two functions, the DE algorithm is better than the LS1 algorithm at only certain stages of the evolution and only when allowed to execute for more evaluations than the ones used for the comparison of the algorithms (1250 in this case). If selected at these stages, the DE algorithm is able to reach better regions of the search space that could solve premature convergence problems or accelerate the convergence to the global optimum. Since the initial data does not provide any information to predict the long-term benefits of selecting an algorithm (due to the length of

[2] As mentioned earlier with a precission of 1E−14.

Table 3. Average Score Values

Function	DE	LS1	baseHRH	randomHRH	smartHRH
f1	6.78E−01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f2	7.71E+01	5.99E+01	1.58E+01	**5.34E+00**	1.34E+01
f3	**2.46E+02**	6.98E+03	8.25E+03	1.26E+03	7.63E+03
f4	1.33E+00	**0.00E+00**	4.78E−01	1.60E+01	**0.00E+00**
f5	1.72E−01	3.25E−03	4.63E−03	9.86E−04	**0.00E+00**
f6	9.77E−02	1.00E−12	1.00E−12	1.00E−12	**0.00E+00**
f7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f8	1.97E+05	4.78E+00	4.54E+00	8.30E+00	**4.52E+00**
f9	**0.00E+00**	5.23E+02	3.97E−06	8.16E+01	**0.00E+00**
f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.06E−08	**0.00E+00**
f11	**0.00E+00**	4.91E+02	8.41E−06	7.89E+01	**0.00E+00**

Table 4. Comparison between algorithms

algorithm	baseHRH	smartHRH	DE	LS1	randomHRH
baseHRH			1,2,4,8	2,9,11	4,8,9,10,11
smartHR	5,6,9,11		1,2,4,8	2,5,6,9,11	4,5,6,8,9,10,11
DE	3,5,6,9,11	3		3,5,6,9,11	3,4,5,6,9,10,11
LS1			1,2,4,8		4,8
randomHRH	2,3,5,6,8	2,3	1,2	2,3,5,9,11	

the period of evaluations), no record of the DE algorithm is generated and no transition to this algorithm is made in the smartHRH algorithm.

In the remaining functions, the smartHRH algorithm is able to detect the best strategy for obtaining the best results. For some functions, the smartHRH decides to execute only one of the algorithms (the best one for that function) whereas for others it combines them in order to obtain more stable results or to reach better solutions. For example, in f5, all the algorithms reach the global optimum in some of their executions, whereas the smartHRH algorithm is the only one to achieve this goal in all of its executions. In f6, the combination of both algorithms allow the smartHRH algorithm to reach the global optimum in all of its executions whereas none of the other algorithms is able to reach it in any of its executions. An example of the evolution of the score of a single execution of the algorithms over the f6 function is displayed in Figure 2. The DE algorithm is not displayed because it quickly converges to poor regions of the search space. It can be seen that the smartHRH algorithm has discovered a beneficial pattern (executing the DE algorithm after several evaluations of the LS1 algorithm) which allows it to reach the global optimum.

An example of the rules generated by the algorithm for functions f5 and f6 is presented in Table 5. As mentioned before, for f6, the algorithm extracts a simple pattern based on the number of evaluations which allows it to reach the global optimum in all the executions. In f5, the evolution of the algorithms is not

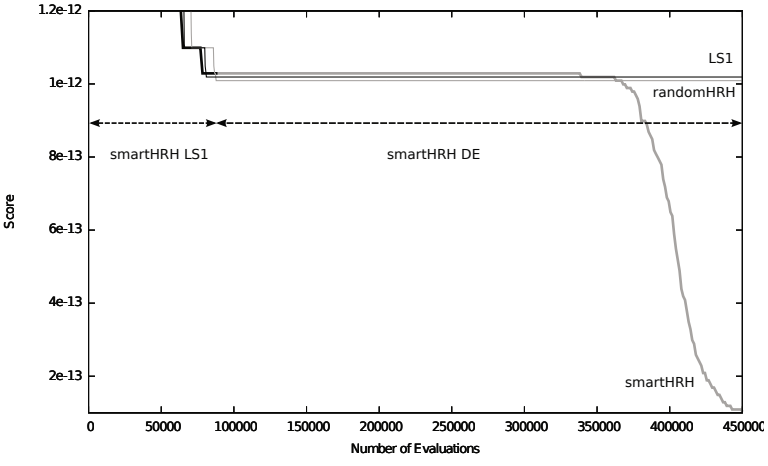


Fig. 2. Comparison of the evolution of the score

Table 5. Rules obtained for functions f5 and f6

f5				
conditions		algorithm	support	precision
if $\#activations_{de}$	<= 9 and			
age_{max}	<= 1899 and			
$avgnevals_{ls1}$	<= 68126	ls1	600	0.99
else if $avgnevals_{ls1}$	<= 37662	de	169	0.93
else if $\#activations_{ls1}$	<= 2 and			
$score_{max}$	<= 0.99	de	64	0.64
else		ls1	65	0.83
f6				
conditions		algorithm	support	precission
if $\#evaluations$	<= 88925	ls1	809	1.0
else		de	358	0.99

always the same and has different patterns. For this reason, the induced model has more rules of higher complexity.

6 Conclusions

In this work, a new hybrid algorithm that learns the best sequence of algorithms has been presented. This learning process uses the information of several parameters of the population, the hybridization and the performance of the algorithms in order to determine the future hybridization strategy. For the experimentation, the benchmark from the workshop on continuous optimization of the ISDA 2009 Conference has been considered. The results have been analyzed and compared with statistical tests. The analysis has proven that the new algorithm is able

to obtain the best overall results, reaching the global optimum in 9 out of 11 functions. This is a first study for validating that the proposed approach can learn very promising hybridization strategies for an HRH algorithm over a set of well-known functions. It must be taken into account that the objective of this study is not to compete against the best algorithms on continuous optimization, since it would not be fair due to the extra number of evaluations used in the learning phase and the per function tuning of the proposed algorithm. As future work we plan to apply this approach to scenarios in which a slightly different optimization problem needs to be solved multiple times. Therefore, the smartHRH algorithm could be trained with several instances of the problem so the resultant algorithm could obtain better results on unseen instances.

Acknowledgments. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa) and the Spanish Supercomputing Network. This work was supported by the Madrid Regional Education Ministry and the European Social Fund and financed by the Spanish Ministry of Science TIN2007- 67148.

References

1. Caponio, A., Neri, F., Tirronen, V.: Super-fit control adaptation in memetic differential evolution frameworks. Soft Computing - A Fusion of Foundations, Methodologies and Applications 13(8), 811–831 (2009)
2. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 16, 122–128 (1986)
3. Hendtlass, T.: A combined swarm differential evolution algorithm for optimization problems. In: Monostori, L., Váncza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS (LNAI), vol. 2070, pp. 11–18. Springer, Heidelberg (2001)
4. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Proc. of Mendel 2000, pp. 76–83 (2000)
5. Muelas, S., LaTorre, A., Peña, J.M.: A memetic differential evolution algorithm for continuous optimization. In: Proc. of ISDA 2009 (November 2009)
6. Ong, Y.-S., Keane, A.: Meta-lamarckian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8(2), 99–110 (2004)
7. Talbi, E.-G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)
8. Tang, K., Yao, X., Suganthan, P., MacNish, C., Chen, Y., Chen, C., Yang, Z.: Benchmark functions for the cec 2008 special session and competition on large scale global optimization. Technical report, USTC (2007)
9. Thangaraj, R., Pant, M., Abraham, A., Badr, Y.: Hybrid evolutionary algorithm for solving global optimization problems. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 310–318. Springer, Heidelberg (2009)
10. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: An enhanced memetic differential evolution in filter design for defect detection in paper production. Evolutionary Computation 16(4), 529–555 (2008)
11. Tseng, L., Chen, C.: Multiple trajectory search for large scale global optimization. In: Proc. of IEEE CEC 2008, June 2008, pp. 3052–3059 (2008)

Gaussian Adaptation Revisited – An Entropic View on Covariance Matrix Adaptation

Christian L. Müller and Ivo F. Sbalzarini

Institute of Theoretical Computer Science and Swiss Institute of Bioinformatics,
ETH Zurich, CH-8092 Zurich, Switzerland
christian.mueller@inf.ethz.ch, ivos@ethz.ch
http://www.mosaic.ethz.ch

Abstract. We revisit Gaussian Adaptation (GaA), a black-box optimizer for discrete and continuous problems that has been developed in the late 1960's. This largely neglected search heuristic shares several interesting features with the well-known Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and with Simulated Annealing (SA). GaA samples single candidate solutions from a multivariate normal distribution and continuously adapts its first and second moments (mean and covariance) such as to maximize the entropy of the search distribution. Sample-point selection is controlled by a monotonically decreasing acceptance threshold, reminiscent of the cooling schedule in SA. We describe the theoretical foundations of GaA and analyze some key features of this algorithm. We empirically show that GaA converges log-linearly on the sphere function and analyze its behavior on selected non-convex test functions.

Keywords: Gaussian Adaptation, Entropy, Covariance Matrix Adaptation, Evolution Strategy, Black-Box Optimization.

1 Introduction

High-dimensional, non-convex, and noisy optimization problems are commonplace in many areas of science and engineering. In many cases, the applied search algorithms have to operate in a black-box scenario, where only zeroth-order information about the objective is available. Such problems can usually only be tackled by stochastic search heuristics, such as Simulated Annealing (SA) [1] or Evolutionary Algorithms (EA). For non-convex, real-valued objective functions, Evolution Strategies (ES), a subclass of EA's, are nowadays the preferred optimization paradigm. A particularly successful example is the Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [2].

In the present paper we revisit Gaussian Adaptation (GaA), a stochastic design-centering and optimization method for discrete and continuous problems that has been introduced and developed since the late 1960s by Gregor Kjellström [3,4]. Although the method shares several interesting features with CMA-ES and with Simulated Annealing, it has been largely neglected in the optimization literature. It is the scope of this work to reintegrate GaA into the field of optimization, as it builds on theoretical concepts that might prove valuable also for other search heuristics. We hereby focus on GaA for continuous sampling and optimization.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 432–441, 2010.

Gaussian Adaptation is a stochastic black-box optimizer that works on discontinuous and noisy functions, where gradients or higher-order derivatives may not exist or are not available. During exploration of the search space, GaA samples *single* candidate solutions from a multivariate normal distribution and iteratively updates the first and second moments of the sampling distribution. While the selection mechanism and moment adaptation of CMA-ES are intended to increase the likelihood of sampling better candidate solutions, GaA adapts the moments such as to maximize the entropy of the search distribution under the constraint that acceptable search points are found with a predefined, fixed hitting (success) probability. If minimization of an objective function is considered, sample-point selection (acceptance) is controlled by a monotonically decreasing, fitness-dependent threshold, reminiscent of the cooling schedule in SA. This ensures that the algorithm focuses on regions with better fitness values.

In order to facilitate understanding of the GaA algorithm and to highlight the key differences in the mean and covariance matrix adaptation strategies, we briefly review several variants of CMA-ES in the following section. We then describe the theoretical foundations of GaA, the algorithmic flow, and the strategy parameter settings. Section 3 illustrates the convergence of GaA on the sphere function and its dependence on search space dimensionality. In order to demonstrate the efficiency of the covariance matrix update, we also report convergence results on Rosenbrock's function. We conclude Section 3 by sketching the maximum entropy behavior of GaA using a test function introduced by Kjellström. Section 4 discusses the obtained results and concludes this work by formulating some theoretical challenges around GaA.

2 Covariance Matrix Adaptation and Gaussian Adaptation

This section summarizes the key concepts of ES with Covariance Matrix Adaptation. Equipped with these preliminaries we then outline the canonical Gaussian Adaption algorithm as developed by Kjellström and co-workers and propose a general parametrization, constraint handling, and initialization protocol.

2.1 Evolution Strategies with Covariance Matrix Adaptation

Standard CMA-ES [2], [5] is a $(\mu/\mu_{\mathrm{w}},\lambda)$-ES that uses *weighted intermediate recombination*, cumulative step size adaptation, and a combination of rank-μ update and rank-one update for the covariance adaptation [6]. At each iteration of the algorithm, the λ members of the candidate population are sampled from a multivariate normal distribution $\mathcal{N}$ with mean $\mathbf{m} \in \mathbb{R}^n$ and covariance $\mathbf{C} \in \mathbb{R}^{n\times n}$. The sampling radius is controlled by the overall standard deviation (step size) σ. Let $\mathbf{x}_k^{(g)}$ the k^{th} individual at generation g. The new individuals at generation $g+1$ are sampled as:

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}\left(\mathbf{0}, \mathbf{C}^{(g)}\right) \quad k = 1,\ldots,\lambda\,. \tag{1}$$

Selection is done by ranking the λ sampled points in order of ascending fitness and retaining the μ best. This procedure renders the algorithm invariant to strictly monotonic transformation of the objective function. The mean of the sampling distribution

given in Eq. 1 is updated using weighted intermediate recombination of the selected points. The covariance matrix for the next generation is adapted using a combination of rank-μ and rank-one update (see [5] for details). The fundamental objective behind the covariance adaptation scheme is to increase the likelihood of finding good samples in the next generation. In addition, self-adaptation of the step size σ enables the algorithm to explore the search space at different scales. Standard settings for the strategy parameters of CMA-ES have been derived from theoretical and empirical studies (see [6] for a comprehensive summary). The restart variant of CMA-ES with iteratively increasing population size (IPOP-CMA-ES) [7] can be considered a parameter-free CMA-ES.

The (1+1)-variant of CMA-ES establishes a direct link to GaA by combining the classical (1+1)-ES with a rank-one update of the covariance matrix [8]. In the next subsection, we show that Gaussian Adaptation has been designed in a similar spirit, yet grounding its theoretical justification on a different foundation.

2.2 Gaussian Adaptation

Gaussian Adaptation has been developed in the context of electrical network design. There, the key goal is to find an optimal setting of design parameters $\mathbf{x} \in \mathbb{R}^n$, e.g., nominal values of resistances and capacities in an analog network, that fulfill two requirements. First, the parameter settings satisfy the specifications imposed by the engineer, i.e. some (real-valued) objective (or criterion) function $f(\mathbf{x})$ applied to the network output, and second, the nominal values should be robust with respect to intrinsic random variations of the components during operation of the electrical device. Kjellström realized that with increasing network complexity classical optimizers such as conjugate gradients perform poorly, especially when analytical gradients are not readily available or when the objective function is multimodal. He suggested to search the space of valid parameter settings with stochastic methods that only rely on evaluations of the objective function. Starting from an exploration method that can be considered an adaptive random walk through design space [3], he refined his algorithm to what he called Gaussian Adaptation [4].

Before turning to the problem of optimization, Kjellström considered the following simpler situation: Assume that the engineer of an electrical circuit can vary the set of design parameters and can decide whether these settings fulfill a specified criterion or not. How can one describe the set $\mathcal{A} \subset \mathbb{R}^n$ of acceptable solutions in a general and compact manner? Based on Shannon's information theory, Kjellström derived that under the assumption of finite mean $\mathbf{m}$ and covariance $\mathbf{C}$ of the samples, a Gaussian distribution may be used to characterize $\mathcal{A}$ [4]. Although not specifically stated in the original publication, Kjellström applied the maximum entropy principle, developed by Jaynes in 1957 [9]. This principle is a type of statistical inference that gives the least biased estimate possible on the given information. In the case of given mean and covariance information, the Gaussian distribution maximizes the entropy $\mathcal{H}$, and hence is the preferred choice to describe the region of acceptable points. The entropy of a multivariate Gaussian distribution is:

$$\mathcal{H}(\mathcal{N}) = \log\left(\sqrt{(2\pi e)^n \det(\mathbf{C})}\right) , \tag{2}$$

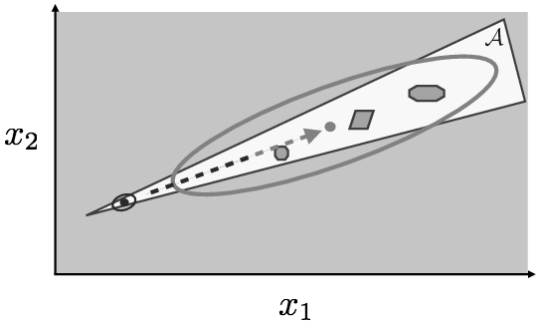


Fig. 1. Illustration of Gaussian Adaptation. The white, non-convex region defines the acceptable region $\mathcal{A}$ in a 2D design-parameter space $\mathbf{x}$. Both the left (dark) and right (light gray) dots and ellipsoids represent the means and covariances of two Gaussian distributions with the same hitting probability P. GaA moves away from the boundary toward the center and adapts the distribution to the shape of $\mathcal{A}$.

where $\mathbf{C}$ is the covariance matrix. In order to get the most informative characterization of the region $\mathcal{A}$, Kjellström envisioned an iterative sampling strategy with a Gaussian distribution that satisfies the following criteria: (i) The probability of finding a feasible design parameter set should be fixed to a predefined value $P < 1$, and (ii) the spread of the samples quantified by their entropy should be maximized. As Eq. 2 shows, this can be achieved by maximizing the determinant of the covariance matrix. In the situation where the parameters have to fulfill a predefined static criterion, the iterative sampler should push the mean of the distribution toward the *center* of the feasible design space. Simultaneously, it should adapt the orientation and scale of the covariance matrix to the shape of $\mathcal{A}$ under the constraint of the fixed hitting probability. The final mean can, e.g., be used as the nominal design parameter set. Fig. 1 illustrates this process, which is called "design centering" or "design tolerancing" in electrical engineering.

When the criterion function $f(\mathbf{x})$ yields real values, the sampler can be turned into a minimizer by introducing a fitness acceptance threshold c_{T} that is monotonically lowered until some convergence criteria are met. A similar idea has later also been employed in the popular Simulated Annealing algorithm [1].

The GaA Algorithm. In order to realize an iterative procedure that works both on design-tolerancing and optimization problems, Kjellström proposed the Gaussian Adaptation method. The process starts by setting the mean $\mathbf{m}^{(0)}$ of a multivariate Gaussian to an initial point $\mathbf{x}^{(\mathbf{0})} \in \mathcal{A}$. The covariance $\mathbf{C}^{(g)}$ is decomposed as follows:

$$\mathbf{C}^{(g)} = \left(r \cdot \mathbf{Q}^{(g)}\right)\left(r \cdot \mathbf{Q}^{(g)}\right)^T = r^2 \left(\mathbf{Q}^{(g)}\right)\left(\mathbf{Q}^{(g)}\right)^T , \tag{3}$$

where r is the scalar step size and $\mathbf{Q}^{(g)}$ is the normalized square root of $\mathbf{C}^{(g)}$. Like in CMA-ES, $\mathbf{Q}^{(g)}$ is found by eigendecomposition of the covariance matrix $\mathbf{C}^{(g)}$. The initial $\mathbf{Q}^{(0)}$ is set to the identity matrix $\mathbf{I}$. A point in iteration $g+1$ is sampled from a Gaussian distribution according to:

$$\mathbf{x}^{(g+1)} = \mathbf{m}^{(g)} + r^{(g)}\mathbf{Q}^{(g)}\eta^{(g)} , \tag{4}$$

where $\eta^{(g)} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$. The new sample is evaluated by the criterion function $f(\mathbf{x}^{(\mathbf{g+1})})$. *Only if* the sample fulfills the specification, i.e. $\mathbf{x}^{(g+1)} \in \mathcal{A}$ in the design-tolerancing scenario or $f(\mathbf{x}^{(g+1)}) < c_{\mathrm{T}}^{(g)}$ in the optimization scenario, the following adaptation rules are applied: The step size r is increased according to $r^{(g+1)} = ss \cdot r^{(g)}$, where $ss > 1$ is called the expansion factor. The mean is updated via

$$\mathbf{m}^{(g+1)} = \left(1 - \frac{1}{N_{\mathrm{m}}}\right) \mathbf{m}^{(g)} + \frac{1}{N_{\mathrm{m}}} \mathbf{x}^{(g+1)} \,. \tag{5}$$

N_{m} is a weighting factor that controls how fast the mean is shifted. The covariance matrix is updated through:

$$\mathbf{C}^{(g+1)} = \left(1 - \frac{1}{N_{\mathrm{C}}}\right) \mathbf{C}^{(g)} + \frac{1}{N_{\mathrm{C}}} \left(\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)}\right) \left(\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)}\right)^T \,. \tag{6}$$

N_{C} weights the influence of the accepted sample point on the covariance adaptation. Kjellström introduced an alternative update rule that is mathematically equivalent to Eq. 6, but numerically more robust. It acts directly on the square root $\mathbf{Q}^{(g)}$ of the covariance matrix:

$$\boldsymbol{\Delta}\mathbf{C}^{(g+1)} = \left(1 - \frac{1}{N_{\mathrm{C}}}\right) \mathbf{I}^{(g)} + \frac{1}{N_{\mathrm{C}}} (\eta^{(g)})(\eta^{(g)})^T, \quad \boldsymbol{\Delta}\mathbf{Q}^{(g+1)} = (\boldsymbol{\Delta}\mathbf{C}^{(g+1)})^{\frac{1}{2}} \,. \tag{7}$$

$\mathbf{Q}^{(g+1)}$ is then updated as $\mathbf{Q}^{(g+1)} = \mathbf{Q}^{(g)} \boldsymbol{\Delta}\mathbf{Q}^{(g+1)}$. In order to decouple the volume of the covariance (controlled by $r^{(g+1)}$) and its orientation, $\mathbf{Q}^{(g+1)}$ is normalized such that $\det(\mathbf{Q}^{(g+1)}) = 1$. As in CMA-ES, the full adaptation of the covariance matrix gives GaA the appealing property of being invariant to arbitrary rotations of the problem.

In case $\mathbf{x}^{(g+1)}$ is not accepted at the current iteration, only the step size is adapted by $r^{(g+1)} = sf \cdot r^{(g)}$, where $sf < 1$ is the contraction factor.

A crucial ingredient for optimization using GaA is the adaptation of the acceptance threshold c_{T}. Kjellström suggested the following rule:

$$\mathbf{c}_{\mathrm{T}}{}^{(g+1)} = \left(1 - \frac{1}{N_{\mathrm{T}}}\right) \mathbf{c}_{\mathrm{T}}{}^{(g)} + \frac{1}{N_{\mathrm{T}}} f(\mathbf{x}^{(g+1)}) \,, \tag{8}$$

where N_{T} controls the weighting between the old threshold and the objective value of the *accepted* sample. It can readily be seen that this fitness-dependent threshold update leaves the algorithm invariant to linear transformations of the objective function.

Strategy Parameters in GaA. Gaussian Adaptation comprises strategy parameters that influence its exploration behavior. We outline a standard parameter setting that is expected to work for a large class of design centering and optimization problems. We first consider the hitting (acceptance) probability P. Kjellström investigated the information-theoretic optimality of P for a random walk in a simplex region [3] and for Gaussian Adaptation in general regions [4]. In both cases, he concluded that the efficiency of the process and P are related as $E \propto -P \log P$, leading to $P = \frac{1}{e} \approx 0.3679$, where e is Euler's number. A proof is provided in [10]. Maintaining the desired hitting probability

corresponds to leaving the volume of the distribution, $\det(\mathbf{C})$, constant under stationary conditions. As $\det(\mathbf{C}) = r^{2n}\det(\mathbf{Q}\mathbf{Q}^T)$, the expansion and contraction factors ss and sf increase or decrease the volume by a factor of ss^{2n} and sf^{2n}, respectively. After S successful and F failed samples, a necessary condition for constant volume thus is:

$$\prod_{i=1}^{S}(ss)^{2n}\prod_{i=1}^{F}(sf)^{2n} = 1\,. \tag{9}$$

Using $P = \frac{S}{S+F}$ and introducing a small $\beta > 0$, one can verify that $ss = 1+\beta(1-P)$ and $sf = 1-\beta P$ satisfies Eq. 9 to first order. The scalar rate β is coupled to the strategy parameters N_C and N_T, but not to N_m. As N_m influences the update of $\mathbf{m} \in \mathbb{R}^n$, it is reasonable to set $N_\mathrm{m} \propto n$. In this study we propose $N_\mathrm{m} = en$. A similar reasoning is employed for N_C and N_T. N_C influences the update of $\mathbf{C} \in \mathbb{R}^{n\times n}$ that contains n^2 entries. Hence, N_C should be proportional to n^2. Kjellström suggests using $N_\mathrm{C} = \frac{(n+1)^2}{\log(n+1)}$ as a standard value, and coupling $N_\mathrm{T} = \frac{N_\mathrm{C}}{2}$ and $\beta = \frac{1}{N_\mathrm{C}}$ [11].

It is noteworthy that the simple (1+1)-ES is a limit case of GaA. Setting $N_\mathrm{m} = N_\mathrm{T} = 1$ moves GaA's mean directly to the accepted sample and c_T to the fitness of the accepted sample. For $N_\mathrm{C} \to \infty$, the covariance stays completely isotropic and GaA becomes equivalent to the (1+1)-ES with a P^{th}-success rule. Keeping N_C finite results in an algorithm that is almost equivalent to the (1+1)-CMA-ES [8]. Slight differences, however, remain in deciding when to update the covariance and how to adapt the step size. Moreover, (1+1)-CMA-ES does not normalize the volume of the covariance matrix. Replacing GaA's acceptance rule by a probability based on Boltzmann-weighted fitness differences, and setting $N_\mathrm{C} \to \infty$, makes GaA equivalent to SA.

Constrained Handling and Initialization. In the context of box-constrained optimization, where the boundaries are explicitly given by $\mathbf{x} \in [\mathbf{A}, \mathbf{B}] \subset \mathbb{R}^n$, several boundary handling techniques can be employed. We suggest projecting the components of samples that violate the constraints onto the boundary along the coordinate axes, and evaluating the projected samples. In the case of box constraints, the initial mean $\mathbf{m}^{(0)}$ is drawn from a uniform distribution in the box. The initial step size is set to $r^{(0)} = 1/e\,(\max\mathbf{B} - \min\mathbf{A})$, similar to the global search setting of the initial σ in CMA-ES [7]. The initial threshold $c_\mathrm{T}^{(0)}$ is set to $f(\mathbf{m}^{(0)})$.

3 Numerical Examples

We show the convergence of GaA on quadratic functions by considering the sphere function. The covariance matrix adaptation mechanism is then demonstrated on Rosenbrock's function. Finally, we sketch the entropic behavior of GaA on a multimodal function introduced by Kjellström [12].

3.1 Gaussian Adaptation on the Sphere Function

We consider the sphere function as a prototypical quadratic function in order to study convergence of the GaA algorithm. It is defined as:

$$f_{\text{Sphere}}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \,. \tag{10}$$

The global minimum is at the origin $\mathbf{0}$ with $f_{\text{Sphere}}(\mathbf{0}) = 0$. For practical purposes, search is restricted to $\mathbf{x} \in [-5, 5]^n$. In order to study the dimension-dependence of GaA's convergence properties, we use the standard strategy parameter values, constraint handling, and initialization. 10 repetitions are conducted for dimensions $n = 2, 5, 10, 20, 30, 40, 50$. Fig. 2 summarizes the results. We observe the expected log-linear convergence of GaA on the sphere function. Nevertheless, we show the results in a log-log plot in order to better discriminate the trajectories for different dimensions (Fig. 2a). The mean number of function evaluations (FES) needed to achieve an accuracy of 10^{-9} grows slightly faster than quadratically with n (Fig. 2b). The measured (empirical) hitting probability $\hat{P}$ approaches the optimal $P = 1/e$ with increasing dimension. A least-squares fit of a power law yields $\hat{P}(n) = -0.2077n^{-0.1831} + 0.4265$ (Inset in Fig. 2b).

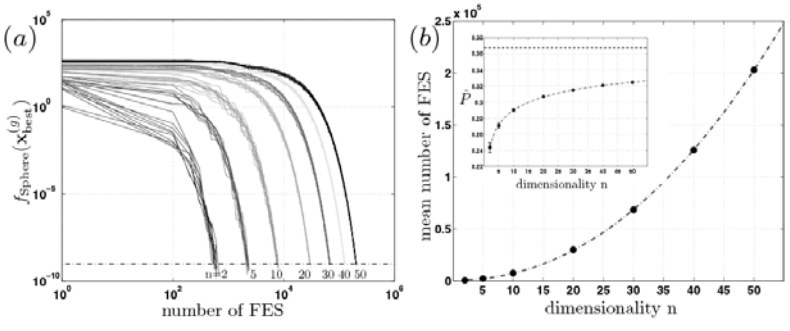


Fig. 2. (a) Log-log plot of the current best fitness value $f(\mathbf{x}_{\text{best}}^{(g)})$ vs. the number of function evaluations (FES) on the sphere function for $n = 2, 5, 10, 20, 30, 40, 50$ (from left to right). The dashed line shows the target fitness (stopping criterion). (b) Average number of FES needed to reach the target fitness vs. dimensionality n. The dashed curve is a perfect fit of the power law $\text{FES}(n) = 47.11n^{2.138} + 857.8$. The inset shows the mean and standard deviation of the empirical hitting probability $\hat{P}$ vs. n. The dashed line represents the desired optimal $P = 1/e$.

3.2 Gaussian Adaptation on Rosenbrock's Function

We study the behavior of the covariance matrix adaptation scheme on Rosenbrock's valley function, defined as:

$$f_{\text{Rosen}}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right) \,. \tag{11}$$

The global minimum is at $\mathbf{1}$ with $f_{\text{Rosen}}(\mathbf{1}) = 0$. Search is constrained to $\mathbf{x} \in [-2, 2]^n$. Rosenbrock's function is multimodal for $n > 3$, and it exhibits an interesting topology. On a global length scale ($\|x_i\| > 1$), the first summand dominates and attracts

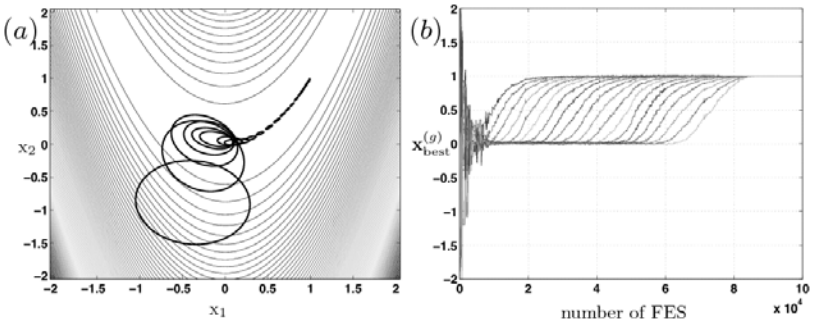


Fig. 3. Typical trajectory of GaA on Rosenbrock's function for $n = 20$. (a) 2D Contour plot with the sub-covariances along the first two dimensions (black ellipses) shown every 1000 iterations. (b) Evolution of the components of $\mathbf{x}_{\text{best}}^{(g)}$ vs. the number of FES (= iterations) for the same run.

most search heuristics toward the origin. On smaller length scales ($\|x_i\| \ll 1$), however, the second term dominates and forms a bent parabolic valley that leads from the origin to the global minimum $\mathbf{1}$. Hence, it becomes favorable to constantly reorient the covariance matrix along the valley. We perform 10 optimization runs with the same protocol for $n = 2, 5, 10, 20, 30, 40$. GaA finds the global minimum in all cases. Similar to CMA-ES, GaA's search on Rosenbrock can be divided into three phases: (1) log-linear convergence toward the origin; (2) a plateau region for covariance adjustment along the valley; (3) log-linear convergence near the global minimum. Fig. 3 shows a typical trajectory of GaA for $n = 20$. The same qualitative behavior is observed also in all other dimensions. After rapidly approaching the origin, GaA efficiently adapts its covariance to follow the valley. The objective variables migrate, in order of increasing dimension, toward the global minimum. The mean number of function evaluations needed to achieve an accuracy of 10^{-9} follows the power law $\text{FES}(n) = 60.13n^{2.462} + 2807$, with offset, prefactor, and exponent being larger than on the sphere function. $\hat{P}$, however, converges toward the optimal value faster than on the sphere function: $\hat{P}(n) = -0.1405n^{-0.917} + 0.3582$.

3.3 Gaussian Adaptation on Kjellström's Function

We consider the highly multimodal test function f_{Kjell} as introduced by Kjellström [12]. This function allows demonstrating under which circumstances a maximum-entropy method such as GaA is effective and efficient. It is defined as:

$$f_{\text{Kjell}}(\mathbf{x}) = \prod_{i=1}^{n}(1 + h(x_i)), \quad h(x_i) = 0.01\sum_{j=1}^{5}[\cos(jx_i + b_j)]\,, \tag{12}$$

with $\mathbf{b} = [b_1, \ldots, b_5] = [1.982, 5.720, 1.621, 0.823, 3.222]$ and $\mathbf{x} \in [0, 2\pi]^n$. The n-dimensional f_{Kjell} is the Cartesian product of n 1D functions. We thus first consider f_{Kjell} in 1D, i.e. $\mathbf{x} = x_1$, as depicted in Fig. 4a. In 1D, f_{Kjell} has 5 minima. The global minimum is at $\mathbf{x}_{\text{min}} \approx 2.3486$ with $f_{\text{Kjell}}(\mathbf{x}_{\text{min}}) \approx 0.9692$. The global maximum $\mathbf{x}_{\text{max}}$

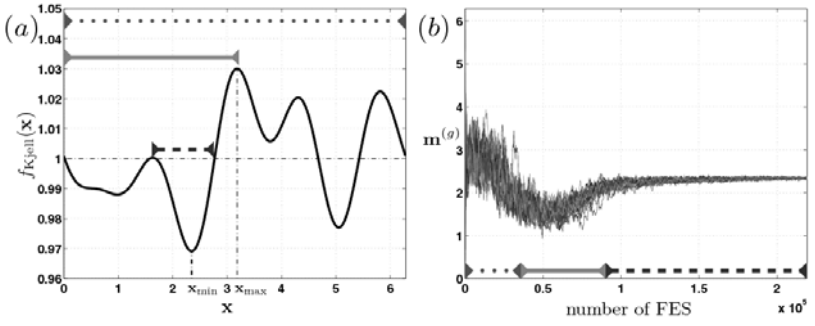


Fig. 4. (a) The multimodal function f_{Kjell} in 1D. The global minimum $\mathbf{x}_{\text{min}}$ is contained in a locally convex region (dashed bar) that belongs to the larger (i.e., maximum entropy) sub-region of the space (solid gray bar). The global maximum $\mathbf{x}_{\text{max}}$ separates this region from the right part of the space. (b) Typical evolution of GaA's mean $\mathbf{m}^{(g)}$ on f_{Kjell} in $n = 25$. After a global search phase (dotted bar), GaA first adapts a high-entropy distribtion to the broad region (solid gray bar) before it converges to the locally convex global-minimum region (dashed bar).

(located at a value slightly larger than $\mathbf{x} = \pi$) divides the search space into two parts. The region $\mathbf{x} < \mathbf{x}_{\text{max}}$ covers a bit more than half of the space and therefore yields higher entropy for an adapted Gaussian distribution (solid gray bar in Fig. 4a). Moreover, it contains, on average, lower function values than the other region, including the global minimum. In n dimensions, f_{Kjell} is a separable multimodal function with 5^n minima. The global minimum is within a region that allows for hierarchical adaptation of Gaussians with high entropy and includes, on average, low objective function values. Despite the staggering number of minima, the function can be solved efficiently. The standard setting for N_{C} (and the coupled parameters N_{T} and β), however, results in premature convergence of the search in one of the $5^{25} \approx 3 \cdot 10^{17}$ minima (success rate $<10\%$). A simple parameter search on N_{C} shows that the setting $N_{\text{C}} = 10n^2$ is better on f_{Kjell} with respect to success rate. It leads to a 100% success rate up to $n = 50$. Fig. 4b depicts the trace of the mean $\mathbf{m}^{(g)}$ for a typical run in $n = 25$ dimensions with optimized N_{C}. In the first phase (dotted interval), GaA explores the entire search space (dotted interval in Fig. 4a). In the second phase (solid interval), it adjusts a high-entropy distribution to the center of the broad region that contains low fitness values (solid interval in Fig. 4a). Finally, GaA proceeds to the region that contains the global minimum with function value $f_{\text{Kjell}}(\mathbf{x}_{\text{min}}) \approx 0.9692^{25} \approx 0.4570$ (dashed interval).

4 Conclusions and Discussion

We have revisited Gaussian Adaptation (GaA), a stochastic design-centering and optimization method that dates back to the 1960's. We have summarized GaA's key concepts, including the maximum entropy principle and the adaptation of the covariance matrix. Furthermore, we have proposed a standard parameter setting, as well as constraint handling and initialization procedures for GaA. We have empirically shown the convergence of GaA on the sphere function and the covariance adaptation mechanism

on Rosenbrock's function. Furthermore, we have re-introduced a highly multimodal function, referred to as Kjellström's function. This function can efficiently be solved by GaA despite the exponentially growing number of minima. This is due to the function's global topology, where the global minimum is located in a region that is suited to maximum-entropy adaptation. Future work will involve a more comprehensive study of GaA's convergence behavior, including a full evaluation on the CEC 2005 benchmark test suite and a convergence proof for quadratic functions. Due to GaA's theoretical foundation on the maximum entropy principle, a framework relating it to modern sampling strategies, such as Adaptive Metropolis or Slice Samplers, is also conceivable.

Acknowledgments. We thank Gregor Kjellström for valuable correspondence, and the TRSH 2009 participants and the anonymous reviewers for their constructive remarks.

References

1. Kirkpatrick, S., Gelatt, C., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
2. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaption in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)
3. Kjellström, G.: Network Optimization by Random Variation of Component Values. Ericsson Technics 25(3), 133–151 (1969)
4. Kjellström, G., Taxen, L.: Stochastic Optimization in System Design. IEEE Trans. Circ. and Syst. 28(7) (July 1981)
5. Hansen, N., Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. 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. 282–291. Springer, Heidelberg (2004)
6. Hansen, N.: The CMA Evolution Strategy: A Tutorial (2007)
7. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proc. of IEEE Congress on Evolutionary Computation (CEC 2005), vol. 2, pp. 1769–1776 (2005)
8. Igel, C., Suttorp, T., Hansen, N.: A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In: GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 453–460. ACM, New York (2006)
9. Jaynes, E.T.: Information Theory and Statistical Mechanics. Phys. Rev. 106(4), 620–630 (1957)
10. Kjellström, G.: On the Efficiency of Gaussian Adaptation. J. Optim. Theory Appl. 71(3) (December 1991)
11. Kjellström, G.: Personal communication
12. Kjellström, G., Taxen, L.: Gaussian Adaptation, an evolution-based efficient global optimizer. In: Comp. Appl. Math., pp. 267–276. Elsevier Science, Amsterdam (1992)

Parallel Genetic Algorithm on the CUDA Architecture

Petr Pospichal, Jiri Jaros, and Josef Schwarz

Brno University of Technology, Faculty of Information Technology, Department of Computer Systems, Bozetechova 2, 612 66 Brno, Czech Republic
Tel.: +420-54114 1364; Fax: +420-541141270
{ipospichal,jarosjir,schwarz}@fit.vutbr.cz

Abstract. This paper deals with the mapping of the parallel island-based genetic algorithm with unidirectional ring migrations to nVidia CUDA software model. The proposed mapping is tested using Rosenbrock's, Griewank's and Michalewicz's benchmark functions. The obtained results indicate that our approach leads to speedups up to seven thousand times higher compared to one CPU thread while maintaining a reasonable results quality. This clearly shows that GPUs have a potential for acceleration of GAs and allow to solve much complex tasks.

1 Introduction

Genetic Algorithms (GA) [3] are powerful, domain-independent search techniques inspired by Darwinian theory. In general, GAs employ selection, mutation, and crossover to generate new search points in a state space. A genetic algorithm starts with a set of individuals that forms a population of the algorithm. Usually, the initial population is generated randomly using a uniform distribution. On every iteration of the algorithm, each individual is evaluated using the fitness function and the termination function is invoked to determine whether the termination criteria have been satisfied. The algorithm ends if acceptable solutions have been found or the computational resources have been spent. Otherwise, the individuals in the population are manipulated by applying different evolutionary operators such as mutation and crossover. Individuals from the previous population are called parents while those created by applying evolutionary operators to the parents are called offsprings. The consecutive process of replacement forms a new population for the next generation.

Although GAs are very effective in solving many practical problems, their execution time can become a limiting factor for some huge problems, because a lot of candidate solutions must be evaluated. Fortunately, the most time-consuming fitness evaluations can be performed independently for each individual in the population using various types of parallelization.

There are different ways of exploiting parallelism in GAs: master-slave models, fine-grained models, island models, and hybrid models [6].

One of the most promising variant is an island model. Island models can fully explore the computing power of course grain parallel computers. The population

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 442–451, 2010.

is divided into a few subpopulations, and each of them evolves separately on different processor. Island populations are free to converge toward different suboptima. The migration operator is supposed to mix good features that emerge locally in the different subpopulations.

Nowadays modern Graphic Processing Units (GPU), although originally designed for real-time 3D rendering can be seen as very fast highly parallel general-purpose systems [4,5] and hence, employed with advantage to accelerate GAs. The second section introduces the main features of nVidia GPU platform. Section 3 describes the mapping of parallel GA onto nVidia CUDA architecture taking into account restrictions of data-parallel processing. Experimental results are presented and discussed in section 4. Section 5 concludes the paper.

2 General Purpose Computation on GPU

Driven by ever increasing requirements from the video game industry, GPUs have evolved into very powerful and flexible processors, while their price remained in the range of consumer market. They now offer floating-point calculation much faster than today's CPU and, beyond graphics applications; they are very well suited to address general problems that can be expressed as data-parallel computations (i.e. the same code is executed on many different data elements).

Moreover, several general purpose high-level languages for GPUs have become available such as CUDA [7] and OpenCL [8] and thus developers do not need any more to master the extra complexity of graphics programming APIs when they design non graphics applications [9].

Modern graphics cards are in fact very powerful massively parallel computers that have (among others) one main drawback: all the elementary processors on the card are organised into larger multi-processors. They have to execute the same instruction at the same time but on different data (SIMD model, for Single Instruction Multiple Data).

GAs need to run an identical evaluation function on different individuals (that can be considered as different data), meaning that this is exactly what GPUs have been designed to deal with. The most basic idea that comes to mind when one wants to parallelize an evolutionary algorithm is to run the evolution engine in a sequential way on some kind of master CPU (potentially the host computer CPU), and when a new generation of offsprings have been created, get them all to evaluate rapidly on a massively parallel computer. This approach has been examined in [12]. The proposed evolutionary algorithm reaches the speedup about 100. But, the bottleneck can be seen in slow data transfers from host memory to GPU and back, especially for small transactions [7].

Another way, how to parallelize GA is to move the whole algorithm on GPU. However, very few researchers so far have gone this way. They usually used Cg language [10,11] which does not allow access to some GPU features (i.e. manual thread and block control). A parallel genetic algorithm targeted to numerical optimization has been published in [13]. Unfortunately, this implementation reached only small speedups between 1.16 and 5.30 depending on population size. Several interesting publication can be also found in [9].

We would like to show, that the movement of entire genetic algorithm can be accomplished in a straightforward way. Moreover, excluding the system bus from the execution, much higher speedups could be achieved.

3 GPU-Based Genetic Algorithm

We have chosen CUDA (Compute Unified Device Architecture) [7] framework to implement our GA on GPU. This toolkit promises best achieved speedups on GPU so far and vast community of developers. CUDA can be performed on any nVidia graphics card from GeForce 8 generation on both Linux and Windows platform. Natural parallelism of computation on GPU is expressed by a few compiler directives added to the well known C programming language.

As mentioned earlier, nVidia GPUs consist of multiprocessors capable to perform tasks in parallel. Threads running in these units are very lightweight and can be synchronized using barriers so that data consistency is maintained. This can be done with very low impact on the performance in a multiprocessor, but not between multiprocessors. This limitation forces us to evolve islands either completely independent or perform migrations between them asynchronously.

The memory attached to graphics cards is divided into two levels — main memory and on-chip memory. Main memory has a big capacity (hundreds of MB) and holds a complete set of data as well as user programs. It also acts as an entry/output point during communication with CPU. Unfortunately, big capacity is outweighed with high latency. On the other hand, on-chip memory is very fast, but has very limited size. Apart from per-thread local registers, on-chip memory contains particularly useful per-multiprocessor shared segments. This 16KB array acts as a user managed L1 cache. The size of on-chip memory is a strongly limiting factor for designing efficient GA, but existing CUDA applications greatly benefit there.

In order to summarize earlier paragraphs, our primary concern during designing GA accelerated by GPU is to create its efficient mapping to CUDA software model with a special focus on massive parallelism, usage of shared memory within multiprocessors and avoiding the system bus bottleneck.

Fig. 1 shows the GA mapping to CUDA software model. We assume an island based GA with the migration along an unidirectional ring. Every individual is controlled by a single CUDA thread. The local populations are stored in shared on-chip memory on particular GPU multiprocessors (CUDA blocks). This ensures both computationally intensive execution and massive parallelism needed for GPU to reach its full potential. Our implementation also utilizes a uniform and Gaussian fast random number generators described in [2].

The proposed algorithm begins with the input population initialization on the CPU side. Then, chromosomes and GA parameters are transferred to the GPU main memory using the system bus. Next, the CUDA kernel performing genetic algorithm on GPU is launched. Depending on kernel parameters, the input population is distributed to several blocks (islands) of threads (individuals). All threads on each island read their chromosomes from the main memory to the fast shared (on-chip) memory within a multiprocessor. From this point, shared

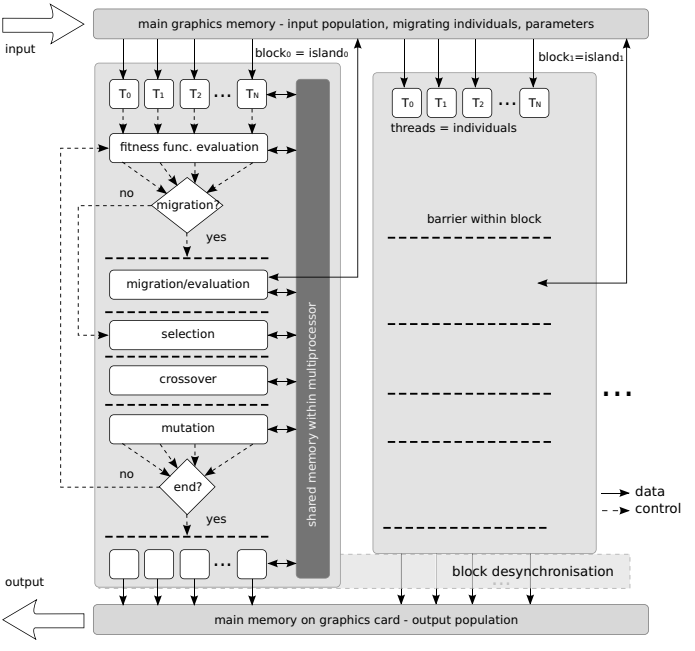


Fig. 1. Mapping of the genetic algorithm to CUDA software model

memories maintain local island populations. The process of evolution then proceeds for a certain number of generations in isolation, whereas, the islands as well as individuals are evolved on the graphics card in parallel. Each generation consists of fitness function evaluation and application of the selection, crossover and mutation (see section 3.1). The operators are separated by CUDA block barriers with zero overhead [7] so that data consistency is ensured.

In order to mix up suitable genetic material from isolated islands, the migration (see section 3.2) is employed. Because migration requires an inter-island communication, slower main memory has to be used for this process. Moreover, since CUDA blocks (islands) cannot be synchronized easily without a significant performance loss, the migration is done asynchronously (it does not wait for the neighbours to complete the predefined number of generations). This is unacceptable for common applications, where data consistency is required, but it works well for stochastic method like GA.

The algorithm iterates until a terminating condition is met (currently the maximum number of generations is set). Finally, every thread writes its evolved chromosome back to the main memory from where it will be read by CPU through the bus.

3.1 Implementation of Genetic Operators

Tournament selection and arithmetic crossover are tightly connected, as it is evident from Fig. 2. Limited shared memory is thereby used efficiently.

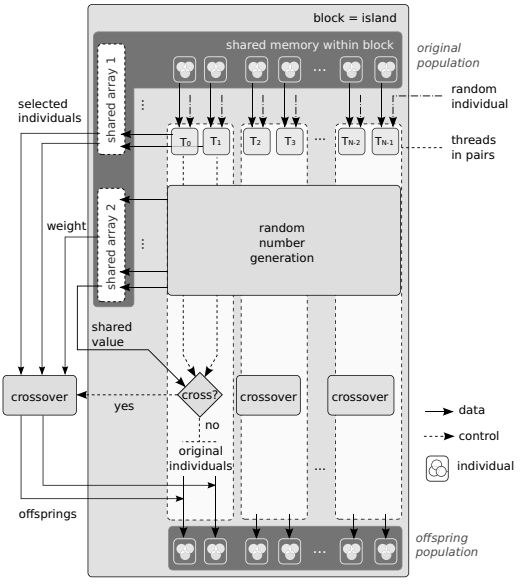


Fig. 2. Scheme of tournament selection with crossover

Threads (individuals) are grouped into pairs using shared variables and barriers so that crossover can be performed in parallel for the whole island population. First, each thread from a pair randomly selects one parent to crossover and it compares the fitness of its own individual using. Then, the index of the better one is written to the shared memory (shared array 1) to notify the other thread in the pair of a more suitable partner to crossover. Next, the parallel uniform random numbers generation is performed in the whole island and the results are written to the shared array 2. Pairs of threads then look up their common random number in this array and compare it with the crossover probability to decide whether perform the crossover or not. This task consumes the first half of the shared array 2. The second half is exploited during the arithmetic crossover as aggregation weights:

$$O_1 = a \cdot P_1 + (1 - a) \cdot P_2 \tag{1}$$

$$O_2 = (1 - a) \cdot P_1 + a \cdot P_2 \tag{2}$$

where O_1 and O_2 represent offsprings, P_1 and P_2 represent parents and a denotes the aggregation weight. This approach wastes the selection of individuals in the case that the crossover is not finally made, but it is from 0.1 to 2% faster (depending on island population size) due to SIMD GPU optimization.

The Gaussian mutation and fitness evaluation are performed in parallel for each thread (see Fig. 1). Finally, the newly generated offsprings replace the parents and the evolutionary cycle is repeated. The elitism is not ensured as crossover may destroy the best chromosome in the island population.

3.2 Migration between Islands

Migration is illustrated in Fig. 3. The islands are interconnected by an unidirectional ring thus an island can only accept individuals from one neighbour. The exchange is done asynchronously using the GPU main memory. The number of migrated individuals is determined by the parameter M. First, the local island population is sorted according to its fitness using Bitonic-Merge sort [1]. Next, M best individuals are written to a part of the main memory belonging to the left neighbour while M worst individuals are overwritten by migrants from a part of the main memory belonging to the right neighbour. Both sorting and migrations are done in parallel for all individuals.

The experimental results show that the migration can significantly improve the convergence to the best solutions in the search space, see table 2.

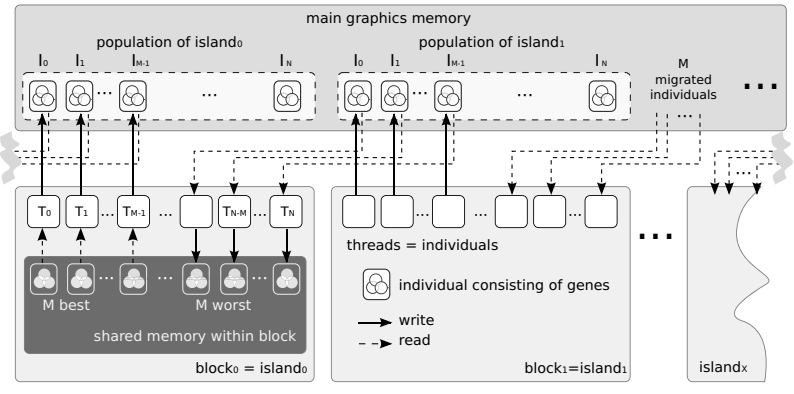


Fig. 3. Scheme of migrations between islands

4 Results

Achievable speedups and solution quality of the proposed GA were examined using Griewank's, Michalewicz's and Rosenbrock's artificial benchmark functions that are often used for GA analysis [15]. CPU version of GA is a single-thread program implemented using well known GAlib library [14].

4.1 Achieved Performance

The speedup of our implementation was investigated using intel Core i7 920 processor and several nVidia consumer-level graphics cards: 8800 GTX (16 multiprocessors / 128 cores), GTX 285 (30 multiprocessors / 240 cores) and GTX 260-SP216 (27 multiprocessors / 216 cores).

The speedup of GPU against CPU were investigated on two dimensional instances of the benchmark functions with mutation probability 5%, crossover rate 70%, no migration and terminating condition of 100 generations. Built-in CUDA timer functions were used to measure GPU kernel execution time [7].

We measured the performance using island population sizes from 2 to 256 individuals, and islands quantity from 1 to 1024. The performance unit was chosen to be population-size independent as IIGG=$\prod$(Island population size, number of Islands, Genotype length, number of Generations) per second. As we expected, CPU performance is almost constant while GPU performance highly varies according to the degree of parallelism (global population size). The performance of graphics cards is also greatly affected by compiler parameter `-use_fast_math` which causes usage of faster, but less precise mathematical functions. This parameter turned out to be profitable for GA because quality of results remains unaffected.

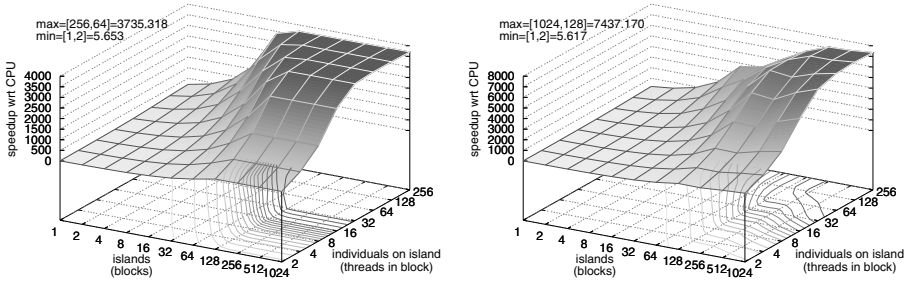
Table 1. Comparsion of CPU and GPU execution performance depending on island population size varying from 2 to 256 individuals. FastMath marks usage of -use_fast_math CUDA compiler parameter.

arch.	**fitness function**	(min – max) **IIGG**·10^6 **per second**		
	Rosenbrock	2.6 – 2.8		
CPU	Michalewicz	1.8 – 2.5		
	Griewank	2.5 – 2.8		
		8800GTX	**GTX260**	**GTX285**
	Rosenbrock	14.2 – 8877	12.0 – 13094	14.3 – 18669
GPU	Rosenbrock-FastMath	18.5 – 11914	15.5 – 17318	18.5 – 24288
	Michalewicz	6.9 – 5893	5.8 – 8850	7.0 – 12937
	Michalewicz-FastMath	11.7 – 9894	9.8 – 13692	11.6 – 19400
	Griewank	9.6 – 7108	8.0 – 10515	9.9 – 14496
	Griewank-FastMath	15.9 – 10507	13.3 – 15360	15.8 – 20920

Table 1 shows measured mean values of IIGG from 5 independent runs to reduce an influence of underlying operating system. GPUs always outperform CPU and truly excel at maximum speeds where measured performance is several thousand times better. Such a huge speedup promises solving problems that took hours to be completed in second-order time.

The results also show that two different generations of GPU, 8800 GTX and GTX285 offer the same performance for the low level of parallelism and differ significantly only at maximum speed. This clearly points out that the new technology is heading for massively-parallel computing rather than improving performance for single threaded applications (see Fig. 4).

The charts shown in Fig. 4 illustrate achieved speedups on two different GPUs against CPU, based on population sizes and the number of simulated island. The 8800 GTX graphic card saturates its computational resources from 256 islands and 32 individuals individuals per an island. The maximal speedup of 3735 against CPU is reached with 256 islands and 64 individuals per island. The GTX285 provides about twice better peak speedup, but it is necessary to provide much more computational work to it. The computational resources of



(a) 8800 GTX (b) GTX 285

Fig. 4. Speedup on Griewank's function depending on GA parameters and GPU

this GPU are not saturated even for 1024 islands and 128 individuals per island, where this GPU has attacked the speedup of 7437[1].

From the both charts in fig. 4 also flow that the usage of the GPU for a small number of island and/or a few individuals per island is not advantageous. The maximum performance is achieved for high number of simulated islands and increasing population size.

4.2 Solutions Quality

The proposed implementation of the tournament selection slightly differs from the original GAlib's one (see section 3.1). In order to ensure the same testing condition for the both CPU and GPU versions, the GAlib's selection were reimplemented. Arithmetic crossover and mutation were kept untouched as they have been defined in the same way.

Tests CPU and GPU1 were performed on artificial benchmark functions mentioned earlier on a single island (obviously with no migrations) with 32 individuals, 70% crossover probability, 5% mutation probability and elitism turned off. Each run was terminated after 100 generations of evolution and the best (lowest) fitness value was taken into consideration.

Test GPU2 was performed with the same GA parameters and benchmarks but with maximum GPU exploitation resulting from simulating 1024 populations (islands) in parallel. Additionally, migrations were performed every 10 generations with 3 individuals (approx. 10% of population).

To compare algorithms adaptability to rising problem complexity, varying number of genes (variables) was tested as well.

Table 2 shows mean value over 100 measured runs. Rosenbrock's and Griewank's functions have the value of the global optimum equal to 0. The value

[1] As it was mentioned earlier, a single threaded CPU implementation was tested. Benchmarked CPU Core i7 allows parallelisation to 4 physical cores + 4 virtual Hyper-Threading ones. Hence, ideally paralleled CPU version with 50% speed benefit from HT technology would change the maximum speedup from 7437 to approx. 1239 times. GAlib also computes variety of additional statistics.

Table 2. Comparison of the solutions quality

genes	mean best fitness								
	Rosenbrock			Michalewicz			Griewank		
	CPU	GPU1	GPU2	CPU	GPU1	GPU2	CPU	GPU1	GPU2
2	0.086	3.468	$7.57 \cdot 10^{-7}$	-1.022	-1.768	-1.801	0.0005	0.0020	$3.99 \cdot 10^{-12}$
3	1.897	4.996	0.447	-1.220	-2.336	-2.760	0.0051	0.0048	$1.06 \cdot 10^{-8}$
4	8.900	4.997	0.494	-1.459	-2.748	-3.696	0.0156	0.0188	$1.22 \cdot 10^{-7}$
5	22.112	17.332	2.042	-1.684	-3.184	-4.628	0.0246	0.0414	0.0001
6	48.450	56.045	4.313	-1.817	-3.654	-5.440	0.0408	0.0570	0.0005
7	83.455	42.509	6.903	-2.035	-3.646	-6.163	0.0479	0.0620	0.0012
8	128.710	155.233	9.257	-2.120	-3.805	-6.659	0.0650	0.1360	0.0027
9	167.329	131.737	12.045	-2.176	-4.830	-7.136	0.0749	0.1444	0.0042
10	233.364	184.370	15.379	-2.391	-5.009	-7.649	0.0805	0.1758	0.0058

of the global optimum of Michalewicz's function varies based on the number of genes – it is approximately linear interpolation from -1.8 (2 genes) to -9.66 (10 genes). Lower value means better solution for all tested functions.

Michalewicz's and Rosenbrock functions are optimised much better on GPU in most cases. On the contrary, Griwank's function for a single island (GPU1) reaches better solutions on CPU. This can be an effect of simplified random number generator which uses very limited amount of shared GPU memory. However, any negative effects are greatly outperformed by massive parallelism. Test GPU2 shows that fully utilized GPU can achieve far better results in the same number of iterations.

Overall, GPU1 results are better than CPU by approx. 20%. This shows that proposed GPU implementation of GA is able to optimise numerical functions.

5 Conclusions

Speedups up to seven thousand times higher clearly show that GPUs have proven their abilities for acceleration of genetic algorithms during optimization of simple numerical functions. The results also show that the proposed GPU implementation of GA can provide better results in the shorter time or produce better results in equal time.

The future work will be oriented to introducing more complex numerical optimization inspired by real-world problems. Moreover, we would like to compare parallel island-based GA running on CPU with the proposed GPU version.

Acknowledgement

This research has been carried out under the financial support of the research grants "Natural Computing on Unconventional Platforms", GP103/10/1517 (2010-2013) of Grant Agency of Czech Republic, and "Security-Oriented Research in Information Technology", MSM 0021630528 (2007-13).

References

1. Pharr, M., Fernando, R.: GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. Addison-Wesley Professional, Reading (2005)
2. Nguyen, H.: GPU gems 3. Addison-Wesley Professional, Reading (2007)
3. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
4. Jiang, C., Snir, M.: Automatic Tuning Matrix Multiplication Performance on Graphics Hardware. In: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques, pp. 185–196 (2005)
5. Galoppo, N., Govindaraju, N.K., Henson, M., Manocha, D.: LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware. In: Proceedings of the ACM/IEEE SC 2005 Conference, vol. 3 (2005)
6. Cant-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Dordrecht (2000)
7. NVIDIA, C.: Compute Unified Device Architecture Programming Guide. NVIDIA: Santa Clara, CA (2007)
8. Munshi, A.: The OpenCL specification version 1.0. Khronos OpenCL Working Group (2009)
9. Harris, M., Luebke, D.: GPGPU: General-purpose computation on graphics hardware. In: Proceedings of the International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2005 Courses, Los Angeles, California (2005)
10. Yu, Q., Chen, C., Pan, Z.: Parallel genetic algorithms on programmable graphics hardware. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 1051–1059. Springer, Heidelberg (2005)
11. Li, J.-M., Wang, X.-J., He, R.-S., Chi, Z.-X.: An efficient fine-grained parallel genetic algorithm based on gpu-accelerated. In: IFIP International Conference on Network and Parallel Computing Workshops, NPC Workshops, pp. 855–862 (2007)
12. Maitre, Q., Baumes, L.A., Lachiche, N., Corma, A., Collet, P.: Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation table of contents, Montreal, Qubec, Canada, pp. 1403–1410 (2009) ISBN 978-1-60558-325-9
13. Wong, M.L., Wong, T.T.: Implementation of Parallel Genetic Algorithms on Graphics Processing Units. In: Intelligent and Evolutionary Systems, vol. 187, pp. 197–216. Springer, Heidelberg (2009)
14. Matthew, W.: GAlib: A C++ Library of Genetic Algorithm Components. Massachusetts Institute of Technology (1996)
15. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Studies in Computational Intelligence. Springer, Heidelberg (2006)

A New Selection Ratio for Large Population Sizes

Fabien Teytaud

TAO (Inria), LRI, UMR 8623 (CNRS - Univ. Paris-Sud), bat 490 Univ. Paris-Sud
91405 Orsay, France
fteytaud@lri.fr

Abstract. Motivated by parallel optimization, we study the Self-Adaptation algorithm for large population sizes. We first show that the current version of this algorithm does not reach the theoretical bounds, then we propose a very simple modification, in the selection part of the evolution process. We show that this simple modification leads to big improvement of the speed-up when the population size is large.

1 Introduction

Evolution Strategies [4,1] are well-known methods for solving an optimization problem. Their main advantages are robustness and simplicity, but, because they are population based they are also well suitable for parallelization. However, it has been shown in [2,6] that ES are not very efficient for large population sizes whereas we can notice that the number of parallel machines, supercomputers and grids, has increased, suggesting big population sizes.

We define λ as the population size, μ as the number of offspring used for the recombination of the parent, and N as the dimensionality. Then, a $(\mu/\mu, \lambda)$-ES (Evolution Strategy) is an evolution strategy with a population size of λ and the μ best offspring will be used, with equal weights, to create the new parent. A $(1, \lambda)$-ES is an evolution strategy in which the new parent is simply the best offspring among the population.

It is widely believed that $(\mu/\mu, \lambda)$-ES are more parallel (efficient for large population sizes) than $(1, \lambda)$-ES, which is shown by rough calculus as in e.g. [1] and by experiments with intermediate values of λ.

Experimentally however, $(\mu/\mu, \lambda)$-ES are less efficient than $(1, \lambda)$-ES for a (sufficiently) large population size λ, for the usual parametrization of the algorithms, as shown in [6]; in particular, the theoretical speedups of ES for parallel optimization (the best possible speed-ups, for optimal algorithms) are far better than the results of current implementations. In this paper, we show that changing the parametrization of self-adaptive $(\mu/\mu, \lambda)$-ES leads to a huge improvement for large λ, and that, with this improvement we can reach the theoretical bounds shown in [7]. These bounds are :

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 452–460, 2010.

- For $(\mu/\mu, \lambda)$-ES, the speed-up is linear until λ of the same order as the dimensionality. For larger value of λ the speed-up becomes logarithmic as a function of λ.
- For $(1, \lambda)$-ES, the speed-up is logarithmic as a function of λ.

Choosing a good selection ratio $\frac{\mu}{\lambda}$ is crucial whatever the dimensionality. Usually, the number of selected individuals μ is function of the population λ; in general, $\mu = \frac{\lambda}{2}$ as suggested in [3] for the covariance matrix adaptation algorithm; $\mu = \frac{\lambda}{4}$ is suggested for the covariance matrix self-adaptation algorithm in [2] which is especially devoted to big population sizes.

We work on the Self-Adaptation Evolution Strategy (SA-ES). It has been proposed in [4] and [5], and extended in [2], in the special case of large λ. This algorithm is very simple and provides one of the state of the art results for large λ. This algorithm is presented in Algorithm 1. $N(0, Id)$ denotes standard multivariate Gaussian random variables with identity covariance matrix.

Algorithm 1. Mutative self-adaptation algorithm. For large population sizes τ is usually equal to $1/\sqrt{N}$; other tuning of τ can been used (e.g. $1/\sqrt{2N}$) which is sometimes found in papers.

Initialize $\sigma^{avg} \in \mathbb{R}$, $y \in \mathbb{R}^N$.
while Halting criterion not fulfilled **do**
  for $i = 1..\lambda$ **do**
    $\sigma_i = \sigma^{avg} e^{\tau N_i(0,1)}$
    $z_i = \sigma_i N_i(0, Id)$
    $y_i = y + z_i$
    $f_i = f(y_i)$
  end for
  Sort the individuals by increasing fitness; $f_{(1)} < f_{(2)} < \cdots < f_{(\lambda)}$.
  $z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z_{(i)}$
  $\sigma^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} \sigma_{(i)}$
  $y = y + z^{avg}$
end while

In this paper, we are considering minimization problems.

2 Discrepancy between Theory and Practice, and the Tuning of the Selection Ratio of the Self-Adaptation Evolution Strategy

In [7], a lot of theoretical bounds have been shown, and, especially that we should reach $\Theta(\log(\lambda))$ with a $(\mu/\mu, \lambda)$-ES. In practice (*i.e.* with usual parametrizations of the algorithm), the SA-ES does not reach that speed-up, as shown in Fig. 1.

In Fig. 1, we can note that the selection ratio μ is a crucial parameter for the behaviour of the SA algorithm. Unless $\mu = 1$, the speed-up reaches a plateau when λ is very large.

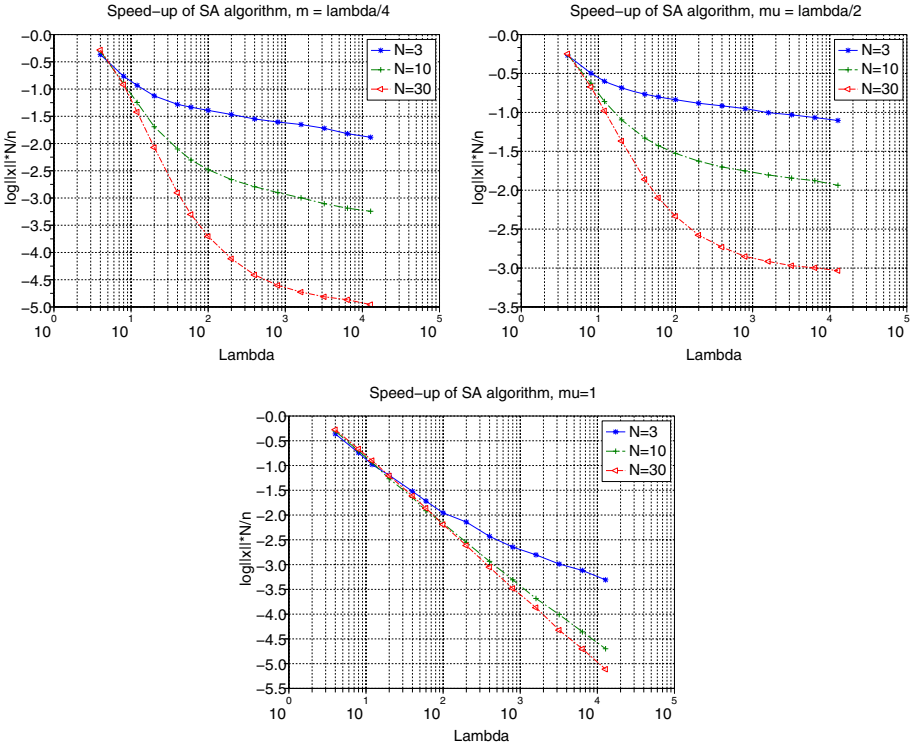


Fig. 1. Speed-up of the Self-Adaptation algorithm as a function of λ, on the sphere function. For this experiment, the initial step size is 1, the initial point is $(1, \ldots, 1)$ and the fitness function to hit is 10^{-10}. Top left ($\mu = \frac{\lambda}{4}$) and top right ($\mu = \frac{\lambda}{2}$) do not show a logarithmic speedup whatever the dimensionality.

$\mu = \frac{\lambda}{2}$ is never a good choice whatever the population size, and if the population size becomes really large, the speedup tends to a constant. $\mu = \frac{\lambda}{4}$ is often used in the literature because it offers a good convergence if the population size is not too large. In fact, Fig. 1 shows that in dimension 10, $\mu = \frac{\lambda}{4}$ is better than $\mu = 1$ until a population size of 200. In dimension 30, $\mu = \frac{\lambda}{4}$ is also a good choice, in particular if the population size is smaller than 6400. The main problem is that the convergence rate tends to a constant (depending on the dimensionality); theoretical results suggest that we should have something better. As said previously, in [7], it is shown that for $(\mu/\mu, \lambda)$-ES algorithms the convergence rate should be linear as a function of λ if the population size is smaller than the dimension. If the population is larger than the dimension the speedup is $\Theta(\log(\lambda))$.

In practice, this is not true. As we can observe in Fig. 1, if the population size is large compared to the dimension, the convergence rate of $(\mu/\mu, \lambda)$-ES tends to

a constant. $(1, \lambda)$-ES only shows a logarithmic convergence rate as a function of λ, which suggests that this is the best choice when the population size is large.

3 Solving the Discrepancy between Theory and Practice by a Good Tuning of SA

In Fig. 2 we show that we can improve the convergence rate by using a simple rule for the selection ratio. Experiments have been done on the sphere function. The initial point is $(1, \ldots, 1)$ and the initial step size is 1. The fitness function to reach is $f_{stop} = 10^{-10}$. In this figure, we compare the Self-Adaptation algorithm

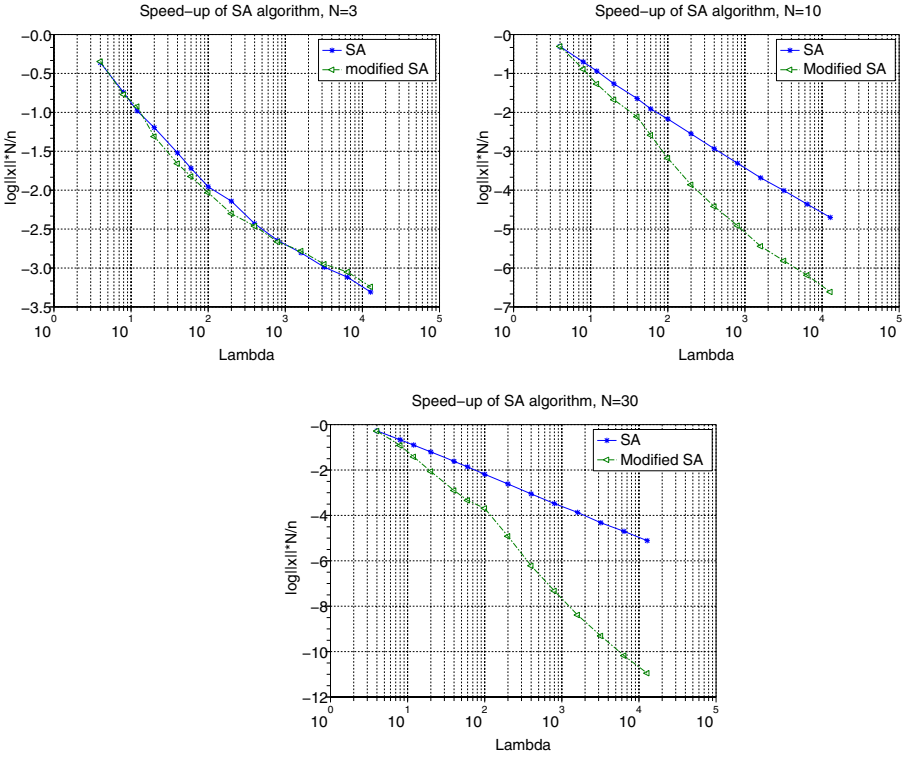


Fig. 2. This figure shows the convergence rate on the sphere function for the Self-Adaptation algorithm and for the modified version of this algorithm. In the modified self adaptation algorithm, the selection ratio is chosen equals to $\mu = \min(N, \lambda/4)$. For the standard self adaptation algorithm, we have chosen the best selection ratio for a large population size ($\mu = 1$). Results are obtained for three dimensions 3, 10 and 30. In dimensions 10 and 30, the modified self adaptation algorithm outperforms the standard version. In dimension 3, results are really close, simply because in that case, the selection ratio of both versions of the algorithm is almost the same (due to the small dimensionality).

with $\mu = 1$ and a modified version of this algorithm. We have chosen $\mu = 1$ for the comparison because it is the best choice for the SA algorithm for large λ, as shown in the previous section.

The modified version is a SA algorithm but with a number of selected points μ equal to the minimum between the dimensionality and the population size divided by 4. Formally, in that case, $\mu = \min(N, \frac{\lambda}{4})$. Intuitively, we want $\mu = N$ for large λ, and $\lambda/4$ for small value of the population size.

In dimension 3, both convergence rates are very close, because the value of the selection ratio of the Self-Adaptation algorithm and of the modified Self-Adaptation algorithm are close to each other.

In dimension 10, the modified version of the Self-Adaptation algorithm outperforms the standard version. We also have a logarithmic convergence rate as predicted by the theory. For a population size of 12800 we have a speedup of 41% over the Self-Adaptation algorithm with $\mu = 1$.

In dimension 30 finally, we have a really big improvement. We have a logarithmic convergence rate, as a function of the population size, and here again results for large population sizes are really good, with a speedup of 114% for a population size of 12800. In this case, for small population sizes, the convergence rate is also slightly better than $\mu = \frac{\lambda}{4}$.

4 Experiments on the Covariance Matrix Self-Adaptation Evolution Strategy

The Covariance Matrix Self-Adaptation Evolution Strategy (CMSA-ES) has been proposed in [2], and is now the state of the art algorithm for large population sizes. This algorithm is based on the SA algorithm presented previously. In the CMSA-ES, the routine used to adapt the global step size σ is the Self-Adaptation one. But we have here a full covariance matrix adaptation algorithm, in order to learn the shape of the fitness function. This algorithm is presented in Algorithm 2.

Following the recommendations in [2] for the tuning of the CMSA-ES algorithm, a selection ratio μ/λ equals to $1/4$ should be used. In this section, we experiment the CMSA evolution strategy with two different selection ratios: the previous one which is recommended, and we also tried our modified version $\mu = \min(N, \lambda/4)$ as above.

4.1 Test Functions

We do our experiments on three well-known functions. Functions are defined in Table 1. The first one is the sphere function, which is a well-known test function in optimization. The second one is the Schwefel Ellipsoid for which the adaptation of the covariance matrix is helpful. The third one is the Rosenbrock function which requires, due to its shape, continuous changes of the covariance matrix.

Algorithm 2. Covariance Matrix self-adaptation. τ is equal to $1/\sqrt{N}$; $< . >$ represents the recombination. The initial covariance matrix C is the identity matrix. The time constant τ_C is equal to $1 + \frac{N(N+1)}{\frac{\lambda}{2}}$.

Initialize $\sigma^{avg} \in \mathbb{R}$, $y \in \mathbb{R}^N$, C.
while Halting criterion not fulfilled **do**
  for $i = 1..\lambda$ **do**
    $\sigma_i = \sigma^{avg} e^{\tau N_i(0,1)}$
    $s_i = \sqrt{C}\sigma_i N_i(0, Id)$
    $z_i = \sigma_i s_i$
    $y_i = y + z_i$
    $f_i = f(y_i)$
  end for
  Sort the individuals by increasing fitness; $f_{(1)} < f_{(2)} < \cdots < f_{(\lambda)}$.
  $z^{avg} = \frac{1}{\mu}\sum_{i=1}^{\mu} z_{(i)}$
  $s^{avg} = \frac{1}{\mu}\sum_{i=1}^{\mu} s_{(i)}$
  $\sigma^{avg} = \frac{1}{\mu}\sum_{i=1}^{\mu} \sigma_{(i)}$
  $y = y + z^{avg}$
  $C = (1 - \frac{1}{\tau_C})C + \frac{1}{\tau_C} < ss^T >$
end while

Table 1. Test functions considered in this paper

Name	Objective function	y_{init}	σ_{init}	f_{stop}
Sphere	$f(y) = \sum_{i=1}^{N} y_i^2$	$(1, \ldots, 1)$	1	10^{-10}
Schwefel	$f(y) = \sum_{i=1}^{N} (\sum_{j=1}^{i} y_i)^2$	$(1, \ldots, 1)$	1	10^{-10}
Rosenbrock	$f(y) = \sum_{i=1}^{N} (100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2)$	$(0, \ldots, 0)$	0.1	10^{-10}

4.2 Results

In our experiments, we have considered different dimensionalities, $N = 3, 10, 30$. We have compared different selection ratios for different population sizes, specially large population sizes. For each point of each experiment we take the average of 60 independent runs. We have chosen to look at the convergence rate. We measure $\frac{N \times log||y||}{number\_of\_Iterations}$ as a function of the population size. Choosing to measure the convergence rate (instead of the number of iterations as in [2]) is here justified by the fact that we know the theoretical limits of this criterion (up to a constant factor), and we know that many usual algorithms have only a bounded speed-up; we want to take care of this.

Fig 3 shows that using a bad selection ratio could be harmful, especially for large population sizes. Even for not so large population sizes (e.g. $\lambda = 20$) using

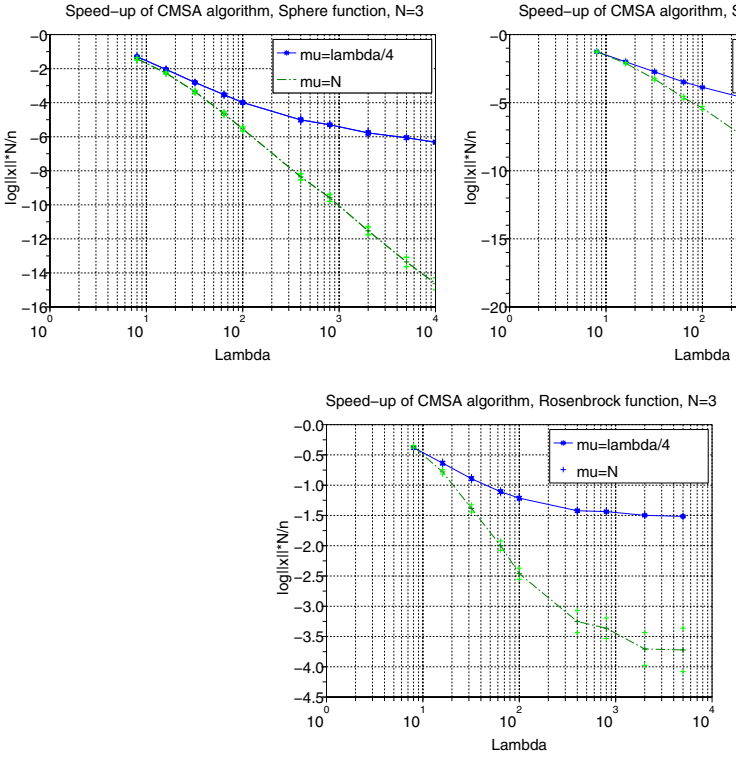


Fig. 3. Results of the CMSA evolution strategy in dimension 3 for the sphere function (top left), the Schwefel function (top right) and the Rosenbrock function. Convergence rate for the standard selection ratio ($\mu = \lambda/4$) and the new selection ratio ($\mu = min(N, \lambda/4)$) are plotted. Choosing a selection ratio equals to $\mu = min(N, \lambda/4)$ is good choice in the three experiments. We reach a logarithmic speedup on the sphere function and the Schwefel function.

the selection ratio $\mu = \min(N, \lambda/4)$ is a good choice. The best speed-ups are reached for large population sizes, more precisely we obtains a speed-up of 136% for the sphere function and 139% for the Schwefel function for a population size equals to 10000 (more than twice faster in both cases), and 146% for the Rosenbrock function for a population size of 5000.

In Fig 4 we experiments the sphere function and the Schwefel function in dimension 10. Due to big computation time, the largest population size for this experiment is 800. Speedup of 37% is reached for the sphere function and 41% for the Schwefel function for a population size of 400.

In Fig 5, we reproduce the experiments as previously, but in bigger dimensionality, 30. Results are similar, and we reach a speedup of 44% for the sphere function and 34% for the Schwefel function for $\lambda = 800$.

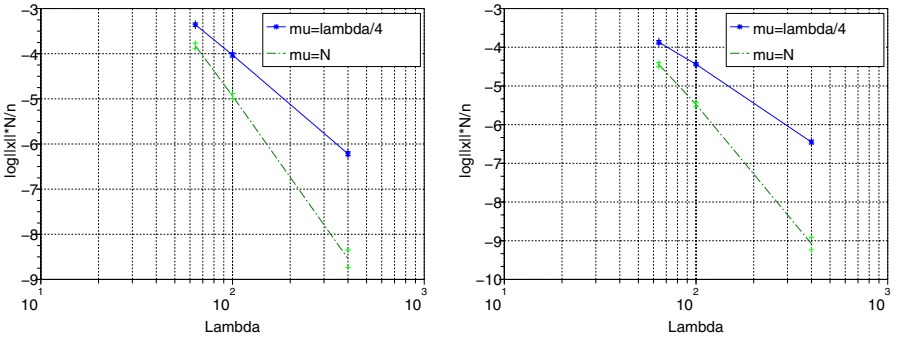


Fig. 4. Results of the CMSA evolution strategy in dimension 10 for the sphere function (top left) and the Schwefel function (top right). Convergence rate for the standard selection ratio ($\mu = \lambda/4$) and the new selection ratio ($\mu = min(N, \lambda/4)$) are plotted.

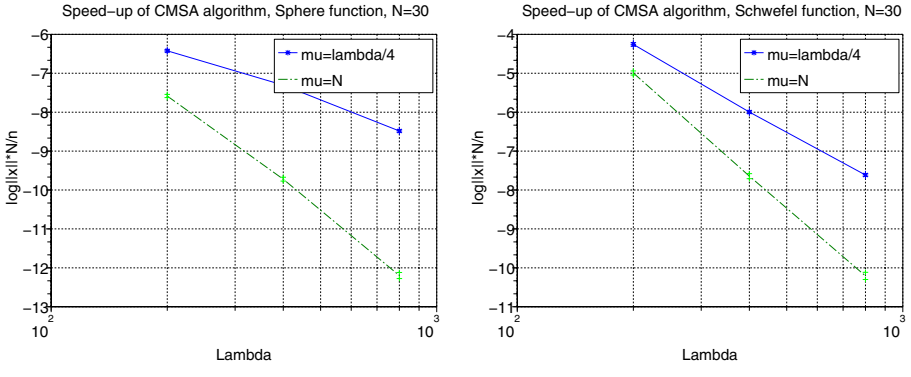


Fig. 5. Results of the CMSA evolution strategy in dimension 3 for the sphere function (top left) and the Schwefel function (top right). Convergence rate for the standard selection ratio ($\mu = \lambda/4$) and the new selection ratio ($\mu = \min(N, \lambda/4)$) are plotted.

5 Conclusion

In this paper we experiment a new rule for the selected population size $\mu = \min(N, \lambda/4)$. This rule is a simple modification, and has very good results for the CMSA evolution strategy. We can seemingly reach a logarithmic speed-up with this rule, which is consistent with the theoretical bounds.

To summarize this paper, we have shown in section 2 that the current version of the Self Adaptation rule only reaches the theoretical speedup (logarithmic as a function of λ) when the selection ratio is equal to $\frac{\mu}{\lambda} = 1/\lambda$, *i.e.* $\mu = 1$.

A selection ratio of $1/2$ or $1/4$ is better at first view, but it's indeed harmful when the population size λ is large enough (larger than the dimension).

In section 3, we have shown that having $\mu = \min(N, \lambda/4)$ could lead to a very good speedup, better than both with $\mu = 1$ and with $\mu = \lambda/4$ or $\mu = \lambda/2$. Finally, we experiment the same rule on the CMSA algorithm (section 4), which is known to be a good evolution strategy when the population size is large.

Unfortunately, due to hard computation time, we only have experimented with small population sizes for dimensions 10 and 30. Experiments with large population sizes should be done in the future. Another future work will be to try this kind of modification on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), *i.e.* in algorithms using cumulative step-length adaptation.

References

1. Beyer, H.G.: The Theory of Evolutions Strategies. Springer, Heidelberg (2001)
2. 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)
3. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
4. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
5. Schwefel, H.P.: Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re 215/3, Technische Universität Berlin (Juli 1974)
6. Teytaud, F., Teytaud, O.: On the parallel speed-up of Estimation of Multivariate Normal Algorithm and Evolution Strategies. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 655–664. Springer, Heidelberg (2009)
7. Teytaud, O., Fournier, H.: Lower bounds for evolution strategies using VC-dimension. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 102–111. Springer, Heidelberg (2008)

Multi-Objective Probability Collectives

Antony Waldock[1] and David Corne[2]

[1] Advanced Technology Centre, BAE Systems, Bristol, UK
antony.waldock@baesystems.com
[2] School of MACS, Heriot-Watt University, Edinburgh, UK
dwcorne@macs.hw.ac.uk

Abstract. We describe and evaluate a multi-objective optimisation (MOO) algorithm that works within the Probability Collectives (PC) optimisation framework. PC is an alternative approach to optimization where the optimization process focusses on finding an ideal distribution over the solution space rather than an ideal solution. We describe one way in which MOO can be done in the PC framework, via using a Pareto-based ranking strategy as a single objective. We partially evaluate this via testing on a number of problems, and compare the results with state of the art alternatives. We find that this first multi-objective probability collectives (MOPC) approach performs competitively, indicating both clear promise, and clear room for improvement.

1 Introduction

1.1 Multi-Objective Optimisation

Multi-objective optimisation (MOO) continues to gain increasing attention [7], as it becomes recognised that (a) a large number of real-world optimisation problems are multi-objective; (b) the classical simplifying approach of combining many objectives into one has several drawbacks [4]; (c) several efficient and effective methods now exist that address MOO in a more principled way (e.g. [32,8,21]).

A MOO problem is formally posed as $\arg\min_{\boldsymbol{x}\in X} G_m(\boldsymbol{x})$, where $G_m(\boldsymbol{x})$ is an objective function and $\boldsymbol{x}$ is defined as a vector of decision variables (or a solution) in the form $\boldsymbol{x} = (x_1, x_2.., x_N)$ from the set of solutions X. The aim is to find the *Pareto* set which contains all the solutions that are not dominated by any other solution. A solution $\boldsymbol{x_1}$ is said to be dominated by $\boldsymbol{x_2}$, if and only if, $\boldsymbol{x_1}$ is as good as $\boldsymbol{x_2}$ in all objectives and $\boldsymbol{x_1}$ is strictly better then $\boldsymbol{x_2}$ in at least one objective. A distinguishing feature of MOO is that the target is a set of solutions rather than a single 'best'.

The most effective MOO approaches to date are generally regarded to be MOEAs (multi-objective evolutionary algorithms). EAs naturally maintain diversity (by working with a population of solutions), and many additional techniques exist (e.g. [11,14,19]). MOEAs encompass a broad family of approaches to MOO; particularly successful among them are those based on particle swarm optimisation (PSO) [17,20,24], and others based on decomposing the MOO problem

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 461–470, 2010.

into several different single-objective problems [16,10]. In the former approach, PSO is combined with the use of measures to maintain a diverse set of 'targets', usually ensuring these are spread well across the developing Pareto set. In the latter approach, the idea is to exploit the relationship between MOO and single objective problems. Roughly speaking, different regions of the Pareto set are optima of different weighted sums of the original objectives; such methods simultaneously progress several different single-objective searches, together aiming to cover the Pareto set.

Meanwhile, an alternative framework for optimisation is Probability Collectives (PC) [3,29], distinguished by a focus on finding an ideal distribution over solution space, rather than an ideal solution. This has similarities to estimation of distribution algorithms (EDAs). However, EDAs operate by continually updating a distribution guided by the results of sampling, with the goal of finding optimal samples. PC, on the other hand, optimizes the distribution itself, and this is reflected in a principled approach to the way that sample evaluations influence the distribution. PC-based optimization is naturally well-suited to maintaining diversity, as well as handling uncertainty and noise. Meanwhile, PC shares with EDAs the clear advantages of using a distribution as the focus of search, rather than just concentrating (and perhaps being misled by) the 'survivors' represented by a current set of samples.

PC clearly has much potential for use in MOO. In this paper, we explore an initial approach to adapting the PC framework for MOO, by formulating the problem using a proxy single-objective, in a similar vein to PSO. It is revealing to see how well this preliminary approach fares against state of the art MOO methods. In the remainder, section 2 introduces Probability Collectives, section 3 introduces the proposed algorithm: Multi-Objective Probability Collectives (MOPC), the experimental set-up and results are in section 4, and we conclude in section 5 with conclusions and future work.

2 Probability Collectives

Probability Collectives (PC) is a framework for black-box optimisation with deep theoretical connections to game theory, statistical physics [26], and optimisation [28]. It has been applied so far to problems, ranging from sensor management [25] to distributed flight control [2].

Typically a single-objective problem is solved by manipulating a vector of decision variables $\boldsymbol{x}$ in an attempt to minimise a scalar function ($G(\boldsymbol{x})$). In PC, however, optimisation is performed on probability distributions $q(\boldsymbol{x})$ over the decision variables, seeking a distribution highly peaked around values of the decision variables that minimise $G(\boldsymbol{x})$. This approach has various demonstrated advantages; it is: easily parallelised (each variable's distribution can be updated independently [18]); applicable to continuous, discrete, or arbitrary spaces [3]; robust to noise and irregularity [9]; provides sensitivity information – a variable with a peaky distribution can be deemed more important than one with a broad distribution. The PC framework formally defines optimisation as in Equation 1.

$$\arg\min_{q_\theta \in \mathscr{P}} \mathbb{E}_{q_\theta}(G(\boldsymbol{x})) \tag{1}$$

where q_θ is a parametric distribution over the decision variables $\boldsymbol{x}$ in the set of all possible distributions $\mathscr{P}$, minimising the expectation $\mathbb{E}_{q_\theta}(G(\boldsymbol{x}))$.

PC does not prescribe a specific approach for minimising this expectation. Many alternatives are discussed in [27]. From hereon, we follow one such approach in which, based on considering the expectation of all possible distributions[22], $\mathbb{E}_{\mathscr{P}}(G(\boldsymbol{x}))$, one solution is the point-wise limit of the Boltzmann distributions shown in Equation 2.

$$p^*(x) = \lim_{\beta \to \infty} p^\beta(\boldsymbol{x}) \tag{2}$$

where $p^\beta(\boldsymbol{x})$ is defined as $\exp[-\beta G(\boldsymbol{x})]$. So, as β tends towards ∞ the distributions of p^β become peaked around the solutions that minimise $G(\boldsymbol{x})$. To find $p^*(\boldsymbol{x})$, a parametrised distribution q_θ is used to approximate the Boltzmann distributions, and fitted to the Boltzmann distribution p^β by minimising the Kullback-Leibler (KL) divergence [13] in Equation 3.

$$\mathbb{E}_{q_\theta}(G(\boldsymbol{x})) = -KL(p^\beta \| q_\theta) \qquad = -\int p^\beta \ln\left(\frac{p^\beta}{q_\theta}\right) dx \tag{3}$$

By minimising the KL Divergence, q_θ will approximate the "target" of $p^\beta(\boldsymbol{x})$. The β term is used as a regularization parameter controlling the evolution of the distribution towards areas of decision space minimising $G(\boldsymbol{x})$. The high-level algorithm we use can now be presented in Alg. 1. We formulate the minimisation

Algorithm 1. PC Optimisation

1. Initialise β to be β_{min}
2. Initialise the number of evaluations to 0
3. **repeat**
4. Draw a set D from X using uniform distribution on the first run or q_θ thereafter
5. Evaluate $G(\boldsymbol{x})$ for each sample drawn
6. Find q_θ by minimising the KL Divergence
7. Update β
8. Update evaluations
9. **until** (evaluations > maximum evaluations)

of KL divergence as cross-entropy minimisation [23] using a single multivariate Gaussian density with mean μ and covariance σ. Means $\boldsymbol{\mu}$ and co-variances $\boldsymbol{\sigma}$ are updated from samples as follows:

$$\boldsymbol{\mu} = \frac{\sum_D s^i \boldsymbol{x}^i}{\sum_D s^i}; \qquad \boldsymbol{\sigma} = \frac{\sum_D s^i (\boldsymbol{x}^i - \boldsymbol{\mu})(\boldsymbol{x}^i - \boldsymbol{\mu})^T}{\sum_D s^i} \tag{4}$$

where s^i is defined as $p(\boldsymbol{x}^i)$ and $\boldsymbol{x}^i$ is the i^{th} sample in the sample set D. Recall that $p(\boldsymbol{x}_i)$ is defined using a Boltzmann distribution $\exp[-\beta G(\boldsymbol{x}^i)]$.

Note that there are close parallels with Expectation Maximisation (EM). The difference from EM is the inclusion of the s^i term which is driven by β included in the Boltzmann distribution. We can regard β as parametrising a trade-off; when small, the parametric distribution tends to fit the samples, regardless of $G(x)$. As β tends towards infinity, the focus shifts towards samples with the best $G(x)$ by producing a highly peaked distribution.

3 Multi-Objective Probability Collectives

Multi-Objective Probability Collectives (MOPC) is implemented here by using a single-objective 'proxy' – i.e. a single-objective score that attempts to evaluate a solution's quality for inclusion in the Pareto set. For this purpose we adopt the maximin function from [17]. There are alternatives, such as Average Ranking or others detailed in [5,1], however our choice of maximin is based on promising performance within a PSO based MOO strategy [17]. Broad pseudocode for MOPC is shown in Alg. 2. In MOPC, β is replaced by T, defined as $\frac{1}{\beta}$ for

Algorithm 2. MOPC Optimisation

1. Initialise the archive set A to empty, T to T_{start} and calculate T_{decay} and the number of evaluations to 0
2. Initialise the set of MOPC Particles P
3. **repeat**
4. **for all** MOPC Particles **do**
5. Update MOPC Particle using A (see algorithm 3)
6. Increment the number of evaluations taken
7. **end for**
8. **if** $(T > T_{end})$ **then** Decrement T **end if**
9. **until** (evaluations > maximum evaluations)
10. Output the non-dominated set from archive set A

convenience. Firstly, MOPC initialises an archive A to keep track of the developing Pareto set, which in our current implementation is maintained in the same way as the Crowding Archive used in NSGAII [8]. Next, MOPC initialises T and a counter for number of evaluations. MOPC calculates the decay rate for T based on T_{start}, T_{end}, the maximum number of evaluations allowed E, the number of particles $|P|$ and the number of samples taken on each iteration $|D|$.

$$T_{decay} = \frac{T_{end}}{T_{start}}^{\frac{|P|*|D|}{E}} \tag{5}$$

MOPC then repeatedly updates each of the particle's parametric distributions while reducing T until the maximum number of evaluations is reached. The output of MOPC is the archive A. Details of line 7 of Alg. 2 are given by Alg. 3.

Algorithm 3. Update MOPC Particle

1. **if** first run **then**
2. Draw and evaluate a set of samples D from X using a uniform distribution for the first run and q_θ thereafter
3. **end if**
4. Add the samples taken in D to a local cache L
5. Calculate maximin for the members of $L \cup A$
6. Find the new q_θ (using L) by minimising the KL Divergence (Eqs. 3, 4)
7. Add the samples from D that are not dominated to the archive A

In Alg. 3, each particle performs its own PC optimisation. First, samples are drawn from q_θ (initially, a uniform distribution), and then added to local cache L to enable previously used samples to be reused when upating the parametric distribution; members of $L \cup A$ are then evaluated using $f_{maximin}$ (Eqn 6):

$$f_{maximin}(x) = \max_{j=1,2..|L\cup A|;x\neq x^j}(\min_{m=1,..,M}(G_i(x) - G_i(x^j))) \tag{6}$$

where m is the objective function, x^j is the j^{th} sample in the set and $f_{maximin}(x)$ is the fitness value for the sample x. This returns a value indicating how much x is dominated by the other samples. When $f_{maximin}(x)$ is $< 0| = 0| > 0$, x is non-dominated | weakly-dominated | dominated. Hence, the distribution is driven towards non-dominated solutions. Distribution q_θ is the updated, and finally, non-dominated samples from D are archived.

4 Experiments and Results

We first explore the behaviour of MOPC on the DEB2 problem in [6], which was specificially designed to present difficult challenges for MOO, arising from multimodality (multiple suboptimal Pareto sets) and deception (structures in the search space that tend to mislead optimisation algorithms towards non-optimal areas). We then consider the set of unconstrained 2-objective problems from the CEC 2009 competition [31]. The DEB2 problem is defined by:

$$minimise\, G_1(x_1, x_2) = x1 \qquad minimise\, G_2(x_1, x_2) \qquad = \frac{g(x_2)}{x_1} \tag{7}$$

where $g(x_2) = 2.0 - \exp\left(-\left(\frac{x_2-0.2}{0.004}\right)^2\right) - 0.8\exp\left(-\left(\frac{x_2-0.6}{0.4}\right)^2\right)$ and x_1 and x_2 are defined in the range $[0.1, 1.0]$. DEB2 is convenient for preliminary exploration since it is two-dimensional and so suitable for visualisation. It has a local minimum around $x_2 \approx 0.6$ and a global minimum around $x_2 \approx 0.2$, facilitating comparison of the relative behaviour of MOPC with other methods.

In these experiments, all the algorithms used 35,000 evaluations for DEB2 and 300,000 evaluations for the CEC2009 problems, and we present means and standard deviations over 30 independent runs. MOPC used 50 particles,

$|A| = 100$, $|D| = 20$, $|L| = 400$, $T_{start} = 1$ and $T_{end} = 0.0001$. NSGAII, MOEA/D-DE and SMPSO implementations where taken from the jMetal framework (`jmetal.sourceforge.net/`v.2.2), all with archive sizes of 100 and other parameters were set as defined in the CEC2009 paper for MOEA/D [30] or as default in JMetal.

4.1 Investigating MOPC on the DEB2 Problem

Table 1 gives means and standard deviations of hypervolume, spread and IGD on DEB2. MOPC and SMPSO outperform MOEA/D and NSGAII, approaching the maximum achievable hypervolume (0.82628), while MOEA/D and NSGAII seem prone to fall into the local minima at $x_2 \approx 0.6$. But, SMPSO achieves better spread than MOPC. Recall that MOPC uses the maximin function and hence should prompt a more uniform spread. To explore this, Figure 1 shows the results of single runs of MOPC and NSGAII. MOPC finds the Pareto set in the small region $x_2 \approx 0.2$, and its dynamics can be seen in Figure 2, showing the evolution of the parametric distribution after 1, 10, 20 and 35 (the final) generations.

Initially centred (by definition of the algorithm), from 10 to 20 iterations the distribution slides to the left and down towards the local minima (of weakly non-dominated solutions) where $x_1 = 0.1$ and $x_2 = 0.6$. Over time, it focuses in two areas where x_2 is approx. 0.2 and along the entire length of x_1, which corresponds to the Pareto set, and the area of weakly non-dominated solutions where x_1 is 0.1. This attraction to weakly non-dominated solutions stymies progression of

Table 1. Comparison between MOPC, NSGAII and MOEAD on the DEB2 problem

Measure	MOPC	MOEA/D-DE [15]	NSGAII [8]	SMPSO [21]
HV	0.81956 (0.00317)	0.73897 (0.08384)	0.77543 (0.07794)	0.82331 (0.00003)
IGD	0.05354 (0.04774)	0.15857 (0.13238)	0.09122 (0.12665)	0.01161 (0.00006)
SPREAD	0.43821 (0.15238)	0.87355 (0.14272)	0.44967 (0.08857)	0.10700 (0.01550)

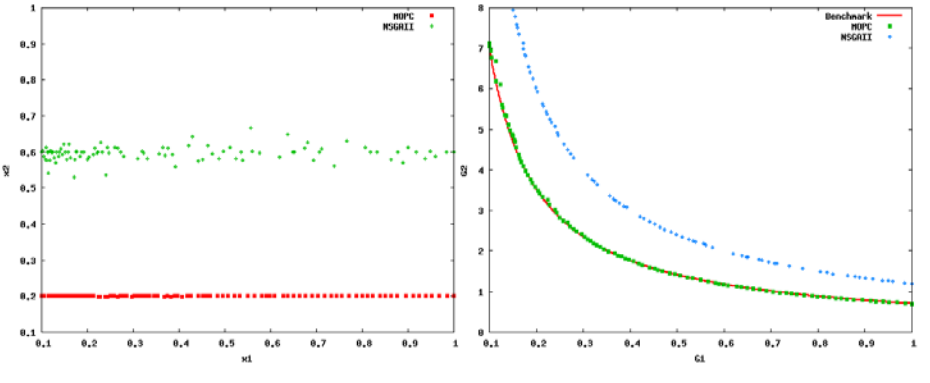


Fig. 1. Decision and objective space for MOPC results on DEB2

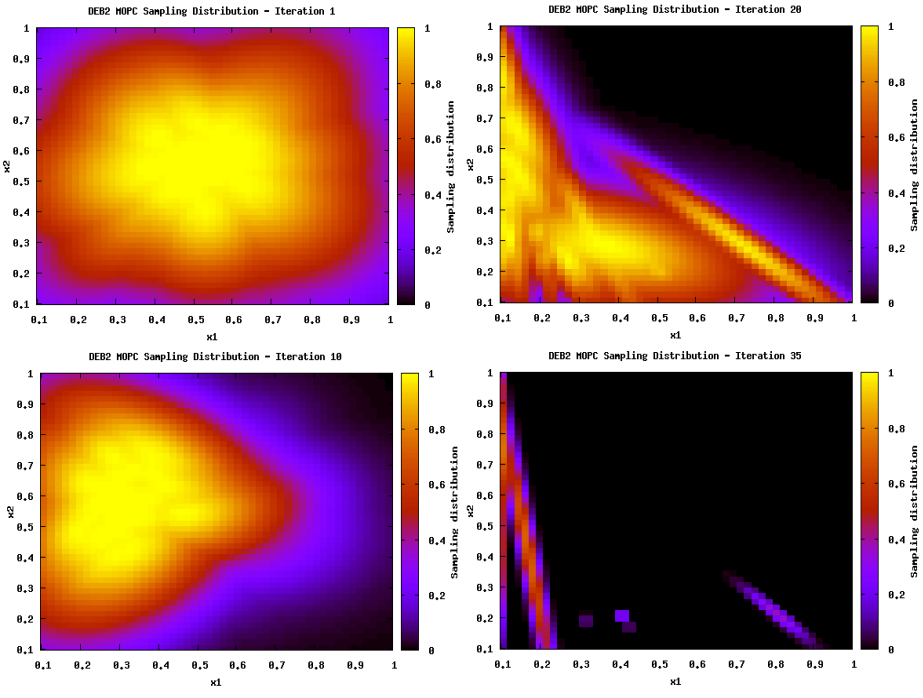


Fig. 2. Parametric distributions of q_θ during optimisation of the DEB2 at iterations 1, 10, 20 and 35

this part of the Pareto set (this can be seen in Figure 1 (b) where the solutions do not exactly match the benchmark set between 0.1 and 0.2).

Overall, the results show that MOPC can find the Pareto set in a small area of the decision space when local minimum exist. Also, MOPC outperforms the MOEA/D and NSGAII implementations tested, but an attraction to weakly non-dominated regions may compromise performance on higher dimensional problems. To investigate this concern, the next section presents experimental results on 30-dimensional problems taken from the CEC 2009 competition [31].

4.2 MOPC Performance on the CEC 2009 Problems

Again, our comparative algorithms are SMPSO [21], MOEA/D [15] and NSGA-II [8]; these are respectvely: an exemplar of the high-performing family of MOEAs based on particle swarm optimisation, the (arguably) state of the art MOEA/D in terms of performance (also an exemplar of the approach to MOO of performing simultaneous single-objective searches in different 'directions', and, finally, the well known accepted benchmark method for MOEAs. They are compared in terms of the IGD metric, which measures the distance from the known Pareto front of the problem in hand. Though not ideal as a comparison metric on MOO [12], this was the chosen metric for the CEC 2009 competition.

Table 2. Comparison of IGD for MOPC, MOEAD, NSGAII and SMPSO on CEC2009

Problem	MOPC	MOEA/D-DE[15]	NSGAII[8]	SMPSO[21]
UF1	0.02417 (0.00355)	0.00855 (0.01438)	0.08803 (0.02916)	0.06279 (0.00663)
UF2	0.03857 (0.00147)	0.03397 (0.00038)	0.03831 (0.00102)	0.04393 (0.00157)
UF3	0.17403 (0.01925)	0.01467 (0.01238)	0.15237 (0.03909)	0.12459 (0.03697)
UF4	0.11505 (0.00605)	0.08440 (0.01070)	0.08314 (0.00356)	0.10830 (0.00463)
UF5	0.50165 (0.02940)	0.73351 (0.11537)	0.37613 (0.07276)	0.74538 (0.15191)
UF6	0.11150 (0.01554)	0.11250 (0.05029)	0.12647 (0.05887)	0.33251 (0.03695)
UF7	0.05208 (0.00297)	0.03880 (0.00020)	0.04878 (0.00280)	0.04810 (0.00139)
UF8	0.36911 (0.01215)	0.39651 (0.10624)	0.15329 (0.00682)	0.23546 (0.01406)
UF9	0.15139 (0.00443)	0.39029 (0.03152)	0.17136 (0.01193)	0.18704 (0.02434)
UF10	0.44892 (0.01813)	0.71918 (0.17908)	0.29066 (0.03195)	0.28438 (0.02143)

The results in Table 2 were produced using the JMetal implementations. We observed that the IGD metric in JMetal was different from that used in the CEC competition, and we note there are several other differences between the jMetal implementation of MOEA/D to the one used in the CEC 2009 competition[30], so, we do not compare our results with those published for the CEC 2009 competition, and consider instead a self-contained comparison among the algorithms described herein.

A broad analysis considering the mean IGD values (lower is better), finds that none of the four algorithms clearly dominates the others. When we consider mean rank of performance over the 10 cases, where 1 is best and 4 is worst, these are respectively 2.2, 2.3, 2.7 and 2.8 for NSGA-II, MOEA/D, MOPC and SMPSO, however this obscures a wide variation in relative performance, Focusing on MOPC, we find its mean is better than MOEA/D in 4 of the 10 cases, better than NSGA-II in 3 of the 10, and better than SMPSO in 5 of the 10. Finally we note there was no attempt to optimise or understand the parameters of MOPC in this initial study.

As a preliminary approach to MOO within the PC framework, it is clear that MOPC indicates some promise for this line of research, showing an ability to outperform, at least in a minority of cases, state of the art MOEAs.

5 Conclusions, Discussion and Future Work

We presented MOPC, a first attempt to design a MOO algorithm within the PC framework. First we explored MOPC on the hard but low-dimensional DEB2 problem, finding that it could approximate the Pareto set well, outperforming NSGAII and MOEA/D, but bettered by SMPSO. We attributed the gap between MOPC and SMPSO to the minimax fitness function, which does not differentiate ideally in the relative selection pressure towards strongly and weakly dominated areas of objective space. On the CEC 2009 problems, MOPC remained promising, outperforming each comparative methods on at least some of the problems. Considering that MOPC is a first approach to MOO within the PC framework,

it is clear that its competitive performance against state of the art MOEAs indicates some promise for the PC strategy in multi-objective optimisation.

Regarding future work, it seems plausible that an approach based on decomposition (as in MOEA/D), is likely to perform well within the PC framework. For example, MOEA/D pursues several single objective optimisation in several well-spread 'directions' at once, where a direction corresponds to a specific weighted sum of the original objectives; this can be done in the PC framework by mapping each such direction to a separate PC-based single-objective optimization. A related highly promising direction is the incorporation of local search, whereby PC provides the global search strategy, but each sample is locally optimzed before the distributions are updated. Finally, a further promising thread is to extend the parametric representation used here to a more complex representation (such as mixtures of multivariate Gaussians) to allow a greater diversity of distributions to be modelled by a single particle.

Acknowledgments

Funded by the Systems Eng. for Autonomous Systems (SEAS) Defence Tech. Centre (DTC). Patent pending - UK application no. 0919343.4.

References

1. Bentley, P., Wakefield, J.: Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms. Springer, Heidelberg (1997)
2. Bieniawski, S.: Distributed Optimization and Flight Control Using Collectives. Thesis, Stanford University (2005)
3. Bieniawski, S., Kroo, I., Wolpert, D.: Discrete, continuous, and constrained optimization using collectives. In: Proc. 10th AIAA/ISSMO M.A.O. Conf., NY (2004)
4. Corne, D., Deb, K., Fleming, P., Knowles, J.: The good of the many outweighs the good of the one. Evol. mult. opt. coNNectionS 1(1), 9–13 (2003)
5. Corne, D., Knowles, J.: Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In: Proc. GECCO 2007 (2007)
6. Deb, K.: Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computation 7, 205–230 (1999)
7. Deb, K.: Multi-Objective Optimization Using EAs. John Wiley and Sons Inc., Chichester (2002)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multi-objective genetic algorithm: Nsga-ii. IEEE Trans on Evol. Comp. 6, 182–197 (2000)
9. Huang, C., Bieniawski, S., Wolpert, D., Strauss, C.: A comparative study of prob. collectives based multi-agent systems and gas. In: GECCO (June 2005)
10. Jaskiewicz, A.: On the perf. of multiple objective genetic local search on the 0/1 knapsack problem: a comparative exp. IEEE Trans. E.C. 6(4), 402–412 (2002)
11. de Jong, E., Watson, R., Pollack, J.: Reducing bloat and promoting diversity using multi-objective methods. In: GECCO, pp. 11–18. MK (2001)
12. Knowles, J., Corne, D.: On metrics for comparing nondominated sets. In: Proc. 2002 Congress on Evolutionary Computation, pp. 711–716 (2002)

13. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. 22(1), 79–86 (1951)
14. Kuo, T., Hwang, S.Y.: Using disruptive selection to maintain diversity in genetic algorithms. Applied Intelligence, 257–267
15. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, moea/d and nsgaii. IEEE Trans. Evolutionary Computation (2008)
16. Li, H., Zhang, Q.: Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comp. 13(2), 229–242 (2009)
17. Li, X.: Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 117–128. Springer, Heidelberg (2004)
18. Macready, W., Wolpert, D.H.: Distributed optimization. In: International Conference on Complex Systems, Boston, MA (May 2004)
19. Morrison, J., Oppacher, F.: Maintaining genetic diversity in genetic algorithms through co-evolution. In: Mercer, R.E. (ed.) Canadian AI 1998. LNCS, vol. 1418, pp. 128–138. Springer, Heidelberg (1998)
20. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In: Proc. of 2008 IEEE Swarm Intelligence Symposium, pp. 26–33. IEEE Service Center, Los Alamitos (2008)
21. Nebro, A., Durillo, J., García-Nieto, J., Coello Coello, C., Luna, F., Alba, E.: Smpso: A new pso-based metaheuristic for multi-objective optimization. In: 2009 IEEE Symp. on MCDM, pp. 66–73. IEEE, Los Alamitos (2009)
22. Rajnarayan, D., Wolpert, D.H., Kroo, I.: Optimization under uncertainty using probability collectives. In: 10th AIAA/ISSMO M.A.O. Conf. (2006)
23. Shore, J., Johnson, R.: Properties of cross-entropy minimization. IEEE Transactions on Information Theory 27(4), 472–482 (1981)
24. Sierra, M.R., Coello, C.A.C.: Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)
25. Waldock, A., Nicholson, D.: Cooperative decentralised data fusion using probability collectives. In: ATSN 2007, at AAMAS 2007, pp. 47–54 (2007)
26. Wolpert, D.H.: Information Theory - the bridge connecting bounded rational game theory and statistical physics (2004)
27. Wolpert, D., Strauss, C., Rajnarayan, D.: Advances in distributed optimization using probability collectives. Advances in Complex Systems 9 (2006)
28. Wolpert, D.: Collective intelligence. In: Fogel, D., Robinson, D. (eds.) Computational Intelligence Beyond 2001: Real and Imagined. Wiley, Chichester (2001)
29. Wolpert, D., Bieniawski, S.: Distributed control by lagrangian steepest descent. In: Proc. 43rd IEEE Conf. on Decision and Control, pp. 1562–1567 (2004)
30. Zhang, Q., Liu, W., Li, H.: The performance of a new version of moea/d on cec09 unconstrained mop test instances. In: CEC 2009: Proceedings of the Eleventh conference on Congress on Evolutionary Computation, pp. 203–208. Institute of Electrical and Electronics Engineers Inc. (2009)
31. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P., Liu, W., Tiwari, S.: MOO test inst. for CEC 09 spec. session and comp. Tech. Rep. CES-887, U. Essex and NTU (2008)
32. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)

Parallel Random Injection Differential Evolution*

Matthieu Weber, Ferrante Neri, and Ville Tirronen

Department of Mathematical Information Technology,
University of Jyväskylä, P.O. Box 35 (Agora), FI-40014, Finland
{matthieu.weber,ferrante.neri,ville.tirronen}@jyu.fi

Abstract. This paper proposes the introduction of a generator of random individuals within the ring topology of a Parallel Differential Evolution algorithm. The generated random individuals are then injected within a sub-population. A crucial point in the proposed study is that a proper balance between migration and random injection can determine the success of a distributed Differential Evolution scheme. An experimental study of this balance is carried out in this paper. Numerical results show that the proposed Parallel Random Injection Differential Evolution seems to be a simple, robust, and efficient algorithm which can be used for various applications. An important finding of this paper is that premature convergence problems due to an excessively frequent migration can be overcome by the injection of random individuals. In this way, the resulting algorithm maintains the high convergence speed properties of a parallel algorithm with high migration but differs in that it is able to continue improving upon the available genotypes and detect high quality solutions.

1 Introduction

Differential Evolution (DE), see [1], is a versatile optimizer which has shown high performance in several applications, especially continuous problems, for example [2]. As highlighted in [3], the success of DE is contained in its implicit self-adaptation which allows an extensive domain exploration during early stages of the evolution and a progressive narrowing of the search within the most promising areas of the decision space. Although this mechanism is effective, it conceals a drawback: the DE contains a limited amount of search moves which could cause the population to fail at enhancing upon the available genotypes, thus resulting in a stagnation condition. In order to overcome this drawback, computer scientists in recent years have attempted to improve the DE performance by modifying the basic DE. Some popular examples of this scientific trend are contained in [4] where multiple search logics are employed, in [5] where the offspring are generated by combining two mating pools (one global and one local,

* This research is supported by the Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 471–480, 2010.

respectively), and in [6] where a randomization of the parameters increases the variability of the potential search moves.

Another popular way to enhance the DE performance is through employment of structured populations. In [7] a distributed DE scheme employing a ring topology (the cores are interconnected in a circle and the migrations occur following the ring) has been proposed for the training of a neural network. In [8], an example of DE parallelization is given for a medical imaging application. A few famous examples of distributed DE are presented in [9] and [10]; in these papers the migration mechanism and the algorithmic parameters are adaptively coordinated according to criterion based on genotypical diversity. In paper [9], a distributed DE that preserves diversity in the niches is proposed in order to solve multi-modal optimization problems. In [11], a distributed DE characterized by a ring topology and the migration of individuals with the best performance, to replace random individuals of the neighbor sub-population, has been proposed. Following similar logic, paper [12] proposes a distributed DE where the computational cores are arranged according to a ring topology and, during migration, the best individual of a sub-population replaces the oldest member of the neighboring population. In [13] a distributed DE has been designed for the image registration problem. In these papers, a computational core acts as a master by collecting the best individuals detected by the various sub-populations running in slave cores. The slave cores are connected in a grid and a migration is arranged among neighbor sub-populations. In [14], a distributed DE which modifies the scheme proposed in [11] has been presented. In [14], the migration is based on a probabilistic criterion depending on five parameters. It is worthwhile mentioning that some parallel implementations of sequential (without structured population) DE are also available in literature, see [15]. An investigation of DE parallelization is given in [16].

This paper focuses on distributed DE and in particular on the ring topology scheme presented in [11]. The main novelty in Parallel Differential Evolution (PDE) described in [11] consists of the migration scheme and its related probability: the DE is independently performed on each sub-population composing the ring and, at the end of each generation, with a certain probability the individual with the best performance is copied in the neighbor sub-population and replaces a randomly selected individual from the target sub-population. In [11] a compromise value of migration probability is proposed. In this paper we propose a novel algorithm based on PDE, namely Parallel Random Injection Differential Evolution (PRIDE). The PRIDE algorithm includes within the ring topology a random generator of candidate solutions: at the end of each generation a random individual replaces a randomly selected individual from a sub-population of the ring topology. This simple mechanism, if properly used, can have a major impact on the algorithmic performance of the original PDE scheme. In addition, the resulting effect of migration (with its probability levels) and random injection is a very interesting topic and has thus been analyzed in this paper.

The remainder of this article is organized in the following way. Section 2 describes the working principles of DE, PDE and PRIDE. Section 3 shows the

experimental setup and numerical results of the present study. Section 4 gives the conclusions of this paper.

2 Parallel Random Injection Differential Evolution

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

At the beginning of the optimization process S_{pop} individuals are pseudo-randomly sampled with a uniform distribution function within the decision space D (for simplicity, the term random will be used instead of pseudo-random in the reminder of this paper). The S_{pop} individuals constituting the populations are distributed over the m sub-populations composing the ring. Each sub-population is composed of $\frac{S_{pop}}{m}$ individuals.

Within each sub-population a standard DE, following its original definition, is performed. At each generation, for each individual x_i of the S_{pop}, three individuals x_r, x_s and x_t are randomly extracted from the population. According to the DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{off} = x_t + F(x_r - x_s) \tag{1}$$

where $F \in [0, 1+[$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_i the offspring should be generated. With $F \in [0, 1+[$, it is meant here that the scale factor should be a positive value which cannot be much greater than 1, see [1]. While there is no theoretical upper limit for F, effective values are rarely greater than 1.0. The mutation scheme shown in Equation (1) is also known as DE/rand/1. It is worth mentioning that there exist many other mutation variants, see [4].

When the provisional offspring has been generated by mutation, each gene of the individual x'_{off} is exchanged with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{off,j} & \text{if } rand(0,1) \leq CR \\ x'_{i,j} & otherwise \end{cases} \tag{2}$$

where $rand(0,1)$ is a random number between 0 and 1; j is the index of the gene under examination.

The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning strategy, it replaces x_i if and only if $f(x_{off}) \leq f(x_i)$; otherwise no replacement occurs. It must be remarked that although the replacement indexes are saved one by one during generation, actual replacements occur all at once at the end of the generation.

According to its original implementation, PDE uses the Parallel Virtual Machine (PVM), allowing multiple computers (called *nodes*) to be organized as a cluster and exchange arbitrary messages. PDE is organized around one master node and m sub-populations running each on one node, and organized as a unidirectional ring. It must be noted that although the logical topology is a ring

which does not contain the master node, the actual topology is a star, where all communications (i.e., the migrations of individuals) are passing through the master.

Each sub-population runs a regular DE algorithm while the master node coordinates the migration of individuals between sub-populations. On each generation, the sub-population has a given probability ϕ to send a copy of its best individual to its next neighbor sub-population in the ring. When migration occurs, the migrating individual replaces a randomly selected individual belonging to the target sub-population.

The PRIDE introduces an apparently minor modification to the PDE scheme. At the end of each generation, according to a certan probability ψ a random individual is generated and inserted within a randomly selected sub-population by replacing a randomly selected individual. However, the replacement cannot involve the individual with the best performance in that sub-population, which is saved in an elitist fashion. For the sake of clarity a scheme highlighting the working principles of PRIDE is shown in Figure 1.

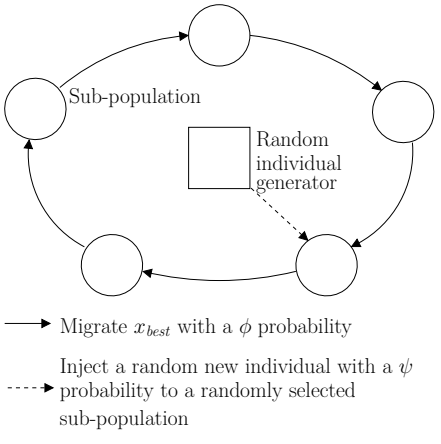


Fig. 1. Working principle of the Parallel Random Injection Differential Evolution

In order to understand the rationale behind the proposed mechanism, it is important to analyze the concept of parallelism and migration in a PDE scheme. As highlighted above, in panmictic DE, stagnation occurs when the algorithm does not manage to improve upon any solution of its population for a prolonged number of generations. In other words, for each stage of the optimization process, DE can generate a limited amount of exploratory moves. If these moves are not enough for generating new promising solutions, the search can be heavily compromised.

Thus, in order to enhance the DE performance, alternative search moves should support the original scheme and promote a successful continuation of the optimization process. The use of multiple populations in distributed DE algorithms allows an observation of the decision space from various perspectives and, most importantly, decreases the risk of stagnation since each sub-population imposes a high exploitation pressure. In addition, the migration mechanism ensures that solutions with a high performance are included within the sub-populations during their evolution. This fact is equivalent to modifying the set of search moves. If the migration privileges the best solutions, the new search moves promote detection of new promising search directions and thus allow the DE search

structure to be periodically "refurbished". Thus, the migration is supposed to mitigate risk of stagnation of the DE sub-populations and to enhance global algorithmic performance.

Migration in PDE has a very interesting effect. Low migration probability values (low values of ϕ) make the algorithm rather explorative, i.e. the PDE slowly improves upon the available solutions and eventually detects individuals characterized by a (relatively) high performance. On the other hand, high migration probability values produce very quick improvements during the early stages of the optimization but eventually cause a premature convergence. The final values detected in these cases may likely have an unsatisfactory performance. This fact can be easily explained as the consequence of a diversity loss due to a high migration: the best individuals are too often copied into the target sub-populations thus causing an excessively aggressive/exploitative algorithmic behavior. In order to obtain an algorithm which produces high quality results with a reasonably good convergence speed, in [11] a compromise on the ϕ value (subsequent to a tuning) has been implemented.

This paper proposes the idea of making use of the highly exploitative behavior of a PDE with a high migration probability, but to inhibit premature convergence by including new genotypes within the ring structure. The algorithmic philosophy employed here is a balance between two opposite and conflicting actions: the migration exploits available search directions by promoting quick improvements upon available genotypes (ϕ action) while the random injection mechanism is supposed to generate completely new genotypes which assist the PDE framework in testing novel promising search directions (ψ action) thus avoiding diversity loss. Analogous to the balance between exploration and exploitation in Evolutionary Algorithms, a proper balance between ϕ and ψ actions determine the success of the proposed PRIDE algorithm.

3 Experimental Results

The test problems listed in Table 1 have been considered in this study.

The rotated version of some of the test problems listed in Table 1 have been included into the benchmark set. These rotated problems have been generated through the multiplication of the vector of variables by a sparse orthogonal rotation matrix, created by composing n rotations of random angles (uniformly sampled in $[-\pi, \pi]$), one around each of the n axes of the search space. The boundaries of rotated problems are kept the same as for the non-rotated problems. In total, ten test problems have been considered in this study with $n = 500$. The choice of the relatively high dimensionality for the test problems in this study is due to the fact that the advantages of PDE-based algorithms with respect to panmictic DE is not evident for low dimensional problems, see [17].

In order to prove the effectiveness of the random injection and its relationship with the migration constant, for each test problem the PDE and PRIDE algorithms have been run with ϕ equal to 0.2 (suggested value in [11]), 0.5, and 1 respectively. On the basis of a preliminary tuning, we decided to set $\psi = 1$

Table 1. Test Problems

Test Problem	Function	Decision Space	Features
Ackley	$-20 + e + 20\exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n} x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)x_i\right)$	$[-1,1]^n$	multimodal, non-separable
Alpine	$\sum_{i=1}^{n} \lvert x_i \sin x_i + 0.1x_i\rvert$	$[-10,10]^n$	multimodal, separable
DeJong's Sphere	$\lVert x\rVert^2$	$[-5.12,5.12]^n$	unimodal, separable
Michalewicz	$-\sum_{i=1}^{n}\sin x_i\left(\sin\left(\frac{i\cdot x_i^2}{\pi}\right)\right)^{20}$	$[0,\pi]^n$	multimodal, separable
Rastrigin	$10n + \sum_{i=0}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right)$	$[-5.12,5.12]^n$	multimodal, separable
Schwefel	$-\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$	multimodal, separable

for all the versions of PRIDE contained in this paper. All the algorithms in this study have been run with populations of 200 individuals divided into 5 subpopulations of 40 individuals each. Regarding scale factor and crossover rate, F has been set equal to 0.7 and CR equal to 0.1 for all the algorithms in this study. Each algorithm has undergone 50 independent runs for each test problem; rotated problems have been rotated using the same matrix on each run. Each single run has been performed for $500,000$ fitness evaluations.

Table 2 shows the average of the final results detected by each algorithm $\pm$ the standard deviations, with the 500 dimension case. Results are organized in PDE vs PRIDE pairs. The best results of each pairwise comparison are highlighted in bold face. The best overall results for each problem are also underlined. With the notation PDE-ϕ and PRIDE-ϕ we mean the PDE and PRIDE algorithms, respectively, running with the corresponding ϕ value. In order to prove statistical significance of the results, the Wilcoxon Rank-sum test has been applied according to the description given in [18] for a confidence level of 0.95. Table 3 shows results of the test. A "+" indicates the case in which PRIDE statistically outperforms its corresponding PDE variant; a "=" indicates that a pairwise comparison leads to success of the Wilcoxon test, i.e., the two algorithms have the same performance; a "-" indicates that PRIDE is outperformed. In order to carry out a numerical comparison of the convergence speed performance for each test problem, the average final fitness value returned by the best performing algorithm (of each pair PDE vs PRIDE under consideration) G has been considered. Subsequently, the average fitness value at the beginning of the optimization process J has also been computed. The threshold value $THR = J - 0.95(J - G)$ has then been calculated. The value THR represents 95% of the decay in the fitness value of the algorithm displaying the best performance. If during a certain run an algorithm succeeds in reaching the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$ required for each algorithm to reach THR has been computed. Subsequently, the Q-test (Q

Table 2. Fitness ± standard deviation

	PDE-0.2	PRIDE-0.2	PDE-0.5
Ackley	$\mathbf{1.62e-01 \pm 1.67e-02}$	$2.86e-01 \pm 1.92e-02$	$1.31e-01 \pm 5.49e-02$
Alpine	$\mathbf{8.88e+01 \pm 1.26e+01}$	$1.22e+02 \pm 1.36e+01$	$8.45e+01 \pm 1.64e+01$
DeJong's Sphere	$\mathbf{1.92e+01 \pm 3.57e+00}$	$4.82e+01 \pm 6.00e+00$	$\mathbf{1.17e+01 \pm 5.17e+00}$
Michalewicz	$\mathbf{-3.06e+02 \pm 5.68e+00}$	$-2.95e+02 \pm 6.50e+00$	$-2.86e+02 \pm 9.56e+00$
Rastrigin	$\mathbf{1.91e+03 \pm 9.94e+01}$	$1.92e+03 \pm 9.04e+01$	$2.24e+03 \pm 1.60e+02$
Schwefel	$-1.30e+05 \pm 3.17e+03$	$\mathbf{-1.32e+05 \pm 2.35e+03}$	$-1.23e+05 \pm 4.11e+03$
Rt. Ackley	$\mathbf{2.15e-01 \pm 2.50e-02}$	$3.67e-01 \pm 3.33e-02$	$\mathbf{1.72e-01 \pm 5.72e-02}$
Rt. Michalewicz	$\mathbf{-1.76e+02 \pm 7.76e+00}$	$-1.76e+02 \pm 5.86e+00$	$-1.52e+02 \pm 7.48e+00$
Rt. Rastrigin	$\mathbf{1.95e+03 \pm 1.51e+02}$	$2.09e+03 \pm 1.06e+02$	$2.34e+03 \pm 1.96e+02$
Rt. Schwefel	$-1.65e+05 \pm 4.74e+03$	$\mathbf{-1.65e+05 \pm 4.78e+03}$	$-1.53e+05 \pm 6.10e+03$

	PRIDE-0.5	PDE-1.0	PRIDE-1.0
Ackley	$\mathbf{1.24e-01 \pm 1.17e-02}$	$2.80e-01 \pm 9.08e-02$	$\mathbf{9.99e-02 \pm 1.10e-02}$
Alpine	$\mathbf{5.56e+01 \pm 7.21e+00}$	$1.59e+02 \pm 2.47e+01$	$\mathbf{4.08e+01 \pm 5.34e+00}$
DeJong's Sphere	$1.32e+01 \pm 2.35e+00$	$4.85e+01 \pm 1.89e+01$	$\mathbf{8.41e+00 \pm 1.70e+00}$
Michalewicz	$\mathbf{-3.55e+02 \pm 5.45e+00}$	$-2.56e+02 \pm 9.29e+00$	$\mathbf{-3.83e+02 \pm 4.21e+00}$
Rastrigin	$\mathbf{1.40e+03 \pm 7.85e+01}$	$2.82e+03 \pm 1.88e+02$	$\mathbf{1.17e+03 \pm 6.07e+01}$
Schwefel	$\mathbf{-1.48e+05 \pm 1.94e+03}$	$-1.12e+05 \pm 4.54e+03$	$\mathbf{-1.57e+05 \pm 1.86e+03}$
Rt. Ackley	$1.95e-01 \pm 2.66e-02$	$3.11e-01 \pm 8.27e-02$	$\mathbf{1.65e-01 \pm 2.40e-02}$
Rt. Michalewicz	$\mathbf{-2.09e+02 \pm 4.57e+00}$	$-1.34e+02 \pm 7.50e+00$	$\mathbf{-2.24e+02 \pm 4.47e+00}$
Rt. Rastrigin	$\mathbf{1.67e+03 \pm 9.08e+01}$	$2.95e+03 \pm 2.32e+02$	$\mathbf{1.51e+03 \pm 6.93e+01}$
Rt. Schwefel	$\mathbf{-1.71e+05 \pm 4.55e+03}$	$-1.36e+05 \pm 6.61e+03$	$\mathbf{-1.74e+05 \pm 4.35e+03}$

Table 3. Wilcoxon Rank-Sum test (PDE vs. corresponding PRIDE)

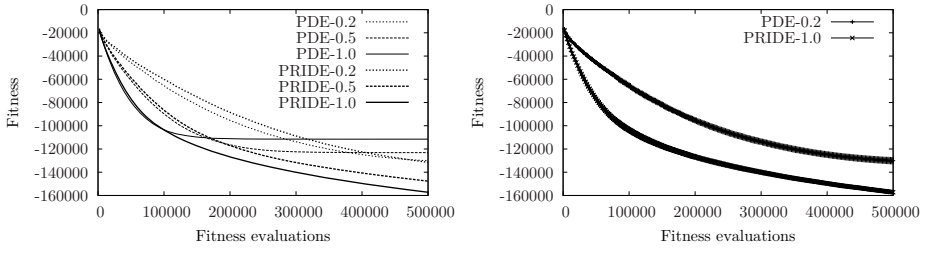
	PDE-0.2	PDE-0.5	PDE-1.0
Ackley	-	=	+
Alpine	-	+	+
DeJong's Sphere	-	-	+
Michalewicz	-	+	+
Rastrigin	=	+	+
Schwefel	+	+	+
Rt. Ackley	-	-	+
Rt. Michalewicz	=	+	+
Rt. Rastrigin	-	+	+
Rt. Schwefel	=	+	+

stands for Quality) described in [3] has been applied. For each test problem and each algorithm, the Q measure is computed as $Q = \frac{\bar{n}e}{R}$ where the robustness R is the percentage of successful runs. It is clear that, for each test problem, the smallest value equals the best performance in terms of convergence speed. The value "∞" means that $R = 0$, i.e., the algorithm never reached the THR. Results of the Q-test are given in Table 4. Best results are highlighted in bold face.

Results can be analyzed from two complementary perspectives: by considering the pairwise comparisons for fixed values of migration constant ϕ and by considering the entire experimental setup. By analyzing the pairwise comparisons, it is clear that in the case $\phi = 0.2$ the random injection does not lead to any improvement; on the contrary in most cases it leads to a worsening of the algorithmic performance and a slowdown in the convergence. In the case $\phi = 0.5$,

Table 4. Q-test

	PDE-0.2	PRIDE-0.2	PDE-0.5	PRIDE-0.5	PDE-1.0	PRIDE-1.0
Ackley	**3.75e+03**	4.59e+03	3.43e+03	**3.29e+03**	7.19e+03	**3.05e+03**
Alpine	**3.84e+03**	4.74e+03	3.33e+03	**3.22e+03**	∞	**2.93e+03**
DeJong's Sphere	**2.28e+03**	2.82e+03	**1.69e+03**	1.92e+03	1.75e+03	**1.63e+03**
Michalewicz	**4.36e+03**	8.51e+03	∞	**4.39e+03**	∞	**4.25e+03**
Rastrigin	**3.74e+03**	4.18e+03	∞	**3.64e+03**	∞	**3.61e+03**
Schwefel	**4.27e+03**	4.30e+03	∞	**4.02e+03**	∞	**4.03e+03**
Rt. Ackley	**3.79e+03**	5.79e+03	**3.40e+03**	3.70e+03	5.60e+03	**3.39e+03**
Rt. Michalewicz	**4.57e+03**	4.84e+03	∞	**4.30e+03**	∞	**4.19e+03**
Rt. Rastrigin	**3.87e+03**	4.60e+03	6.44e+04	**3.61e+03**	∞	**3.49e+03**
Rt. Schwefel	**4.06e+03**	4.43e+03	4.73e+04	**3.91e+03**	∞	**3.92e+03**



(a) (b)

Fig. 2. Performance trend (Schwefel)

employment of random injection is advisable since it succeeds at improving upon the PDE performance in seven cases out of the ten considered in this study. However, not in all cases is this extra component beneficial. Finally in the case $\phi = 1$, there is a clear benefit in the employment of random injection in terms of both final results and convergence speed. In particular Q-test results show that the PDE algorithm in the comparison PDE-1.0 vs PRIDE-1.0 displays many ∞ values, i.e. it is not competitive with respect to its PRIDE variant. This fact can be seen as a confirmation that the introduction of novel (random) individuals within the ring topology can be an efficient countermeasure against premature convergence.

An analysis of the entire experimental setup shows that the PRIDE with $\phi = \psi = 1$ is a very efficient algorithm which outperforms (at least for those tests carried out in this paper) all the other algorithms in this study, in particular the PDE with its most promising parameter tuning ($\phi = 0.2$). More specifically the PRIDE, analogous to the PDE-1.0, tends to reach improvements very quickly during early stages of the evolution, but instead of converging prematurely continues generating solutions with a high performance and eventually detects good results. According to our interpretation, this algorithmic behavior is due to an efficient balance in the exploitation of promising search directions resulting from the frequent migrations and testing of unexplored areas of the decision space resulting from the random injections. Both these components within

a DE structure seem to compensate for the lack of potential search moves and the stagnation inconvenience.

For sake of clarity an example of the performance trend is shown in Figure 2(a). In order to highlight the benefit of random injection, the trend of the average fitness ± standard deviation is shown in Figure 2(b).

4 Conclusion

This paper proposes the generation of random solutions during the evolution of a Parallel Differential Evolution algorithm previously proposed in literature. These random solutions are intended to propose new promising search directions to the sub-populations during the optimization process. An analysis of the algorithmic performance, dependant on the value of migration probability, has been carried out. Experimental results show that while random injections do not lead to benefits for low values of migration probabilities, they could be extremely beneficial in conjunction with high migration probabilities. In particular, an algorithm combining frequent migrations and random injections seems very robust and efficient for various problems and seems to outperform the original Parallel Differential Evolution regardless of its parameter setting. An important finding of this research is the relationship between high migration and random injections which can be seen as parametrization, in relation to Parallel Differential Evolution, of the general concept of balance between exploration and exploitation in Evolutionary Computation.

References

1. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
2. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: An enhanced memetic differential evolution in filter design for defect detection in paper production. Evolutionary Computation 16, 529–555 (2008)
3. Feoktistov, V.: Differential Evolution in Search of Solutions, pp. 83–86. Springer, Heidelberg (2006)
4. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13, 398–417 (2009)
5. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution with a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation 13(3), 526–553 (2009)
6. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
7. Kwedlo, W., Bandurski, K.: A parallel differential evolution algorithm. In: Proceedings of the IEEE International Symposium on Parallel Computing in Electrical Engineering, pp. 319–324 (2006)

8. Salomon, M., Perrin, G.R., Heitz, F., Armspach, J.P.: Parallel differential evolution: Application to 3-d medical image registration. In: Price, K.V., Storn, R.M., Lampinen, J.A. (eds.) Differential Evolution–A Practical Approach to Global Optimization. Natural Computing Series, pp. 353–411. Springer, Heidelberg (2005)
9. Zaharie, D.: Parameter adaptation in differential evolution by controlling the population diversity. In: Petcu, D., et al. (eds.) Proceedings of the International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, pp. 385–397 (2002)
10. Zaharie, D., Petcu, D.: Parallel implementation of multi-population differential evolution. In: Proceedings of the NATO Advanced Research Workshop on Concurrent Information Processing and Computing, pp. 223–232. IOS Press, Amsterdam (2003)
11. Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2023–2029 (2004)
12. Kozlov, K.N., Samsonov, A.M.: New migration scheme for parallel differential evolution. In: Proceedings of the International Conference on Bioinformatics of Genome Regulation and Structure, pp. 141–144 (2006)
13. 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)
14. Apolloni, J., Leguizamón, G., García-Nieto, J., Alba, E.: Island based distributed differential evolution: An experimental study on hybrid testbeds. In: Proceedings of the IEEE International Conference on Hybrid Intelligent Systems, pp. 696–701 (2008)
15. Nipteni, M.S., Valakos, I., Nikolos, I.: An asynchronous parallel differential evolution algorithm. In: Proceedings of the ERCOFTAC Conference on Design Optimisation: Methods and Application (2006)
16. Lampinen, J.: Differential evolution - new naturally parallel approach for engineering design optimization. In: Topping, B.H. (ed.) Developments in Computational Mechanics with High Performance Computing, pp. 217–228. Civil-Comp Press (1999)
17. Weber, M., Neri, F., Tirronen, V.: Distributed differential evolution with explorative-exploitative population families. Genetic Programming and Evolvable Machines 10(4), 343–371 (2009)
18. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)

Effect of Spatial Locality on an Evolutionary Algorithm for Multimodal Optimization

Ka-Chun Wong, Kwong-Sak Leung, and Man-Hon Wong

Department of Computer Science & Engineering
The Chinese University of Hong Kong, HKSAR, China
{kcwong,ksleung,mhwong}@cse.cuhk.edu.hk

Abstract. To explore the effect of spatial locality, crowding differential evolution is incorporated with spatial locality for multimodal optimization. Instead of random trial vector generations, it takes advantages of spatial locality to generate fitter trial vectors. Experiments were conducted to compare the proposed algorithm (CrowdingDE-L) with the state-of-the-art algorithms. Further experiments were also conducted on a real world problem. The experimental results indicate that CrowdingDE-L has a competitive edge over the other algorithms tested.

1 Introduction

Real world problems always have different solutions. Unfortunately, most traditional optimization techniques focus on solving for a single optimal solution. They needs to be applied several times; yet all solutions are not guaranteed to be found. Thus multimodal optimization problem was proposed. In this problem, we are interested in not only a single optimal point, but also the others. Given an objective function, an algorithm is expected to find all optimal points in a single run. With strongly parallel search capability, evolutionary algorithms are shown to be particularly effective in solving this type of problems [7].

2 Background

The work by De Jong [3] is one of the first known attempts to solve multimodal optimization problems by an evolutionary algorithm. He introduced a technique called "crowding": An offspring replaces the most similar individual. As a result, it can maintain different types of individuals in a single run to increase the chance for locating multiple optima. Since then, researchers have proposed different genetic algorithms for multimodal optimization problems. In particular, crowding [3], fitness sharing [8], and speciation [9,19] techniques are the most popular techniques. They have also been integrated in differential evolution [17] and demonstrated promising results [18,10].

Differential Evolution was proposed by Storn and Price [17]. Without loss of generality, a typical strategy of differential evolution (DE/rand/1) [6] is briefly described as follows: For each individual $indiv_i$ in a generation, the algorithm randomly selects three individuals to form a trial vector. One individual forms a base vector, whereas the value difference between the other two individuals forms a difference vector. The sum

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 481–490, 2010.

of these two vector forms a trial vector, which recombines with $indiv_i$ to form an offspring. In a comparison to crossover and mutation operations, it can provide differential evolution a self-organizing ability and high adaptability for choosing suitable step sizes which demonstrated its potential for continuous optimization in the past contests [2].

2.1 CrowdingDE

To extend the capability of differential evolution, Thomsen [18] incorporates the crowding technique [3] into differential evolution (CrowdingDE) for multimodal optimization. Although an intense computation is accompanied, it can effectively transform differential evolution into an algorithm specialized for multimodal optimization. To determine the dissimilarity (or distance) between two individuals, the dissimilarity measurement proposed in Goldberg et al. [8] and Li et al. [9] is adopted. The dissimilarity between two individuals is based on their Euclidean distance. The smaller the distance, the more similar they are and vice versa.

2.2 Locality of Reference

Locality of Reference [15] (or **The Locality Principle** [5]), is one of the most fundamental principles widely used in computing. The principle was originated from memory management methods in order to predict which memory entries would be referenced soon. The main idea is to make use of neighborhood relationships for prediction, optimizing the throughput. To define the neighborhood relationship, time and space are typically taken as the proximity measures. If time is taken, it is called **temporal locality**. If space is taken, it is called **spatial locality**.

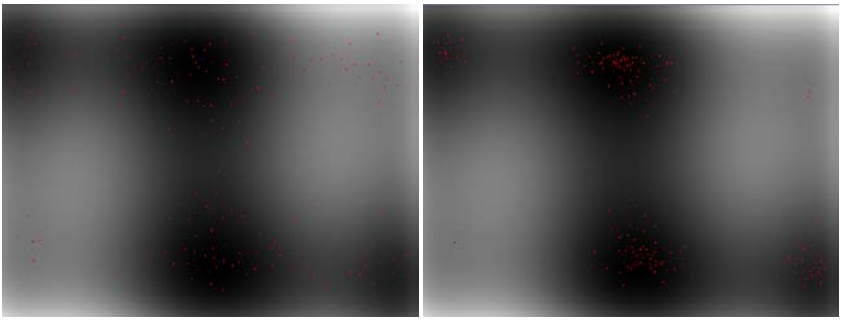
3 Proposed Method

3.1 Motivation

If we do not apply any specific technique to maintain diversity, most evolutionary algorithms will prematurely converge and get stuck in a local optimum. To cope with the problem, the algorithms for multimodal optimization are usually equipped with their own local operations for diversity maintenance. In CorwdingDE, its local operation is the crowding technique. Thinking this technique more deeply, it is to propose a restriction on the individual replacement policy such that an individual gets replaced only when a fitter offspring is generated within the same niche. Thus the choice of the offspring generation method becomes a critical performance factor. Unfortunately, the offspring generations in CrowdingDE mainly relies on random trial vector generations. They are too random to offer enough feasible replacement schemes for all individuals. Thus we propose a new method for trial vector generation, in order to increase the chances for successful replacements.

3.2 CrowdingDE-L

Close individuals tend to have similar characteristics. For instance, two population snapshots of CrowdingDE are shown on Fig. 1. For each snapshot, the population can



(a) 5000 function evaluations (b) 20000 function evaluations

Fig. 1. Population snapshots of CrowdingDE [18] on F5 with population size = 200

be seen to be divided into different niches. Within each niche, the individuals exhibit similar positions and step-sizes for improvement. After several generations, the difference between niches may be even larger. It will be a disaster if a single evolutionary strategy is applied to all of them regardless of their niches. Luckily, it is a two-edged sword. Such property also gives us spatial locality: crossovers between close individuals can have higher chances to generate better trial vectors.

Thus a local operation is proposed to take advantage of it: the individuals which are closer to a parent should be given more chances to be involved in the parent's trial vector generation. In other words, to bring such neighborhood idea into the trial vector generation, the distances between the parent and its candidate individuals are first computed. Then the distances are transformed into the proportions of a roulette-wheel [4]. Within the roulette-wheel, larger proportions of the roulette-wheel are given to closer candidate individuals. It follows that closer individuals are given higher chances for trial vector generations. For the sake of clarity, the local operation is outlined in Algorithm 1.

Combined with the local operation, Crowding Differential Evolution (CrowdingDE) is reformulated as a hybrid algorithm which takes advantages of spatial locality. Thus it is named **Crowding D**ifferential **E**volution using Spatial **L**ocality (**CrowdingDE-L**). A trial vector generation can be tailor-made for each individual. Fitter trial vectors are more likely to be generated. More feasible replacement schemes can thus be provided.

Mathematically, a function is needed to transform distances to proportions of a roulette-wheel. Thus two transformation functions are proposed in this paper: Since closer individuals are given higher values (proportions), the transformation function must be a monotonically decreasing function over the interval [0,MaxDistance], where MaxDistance is the maximum distance between a parent and all of its candidate individuals. Thus a simple function and Gaussian function are proposed for the transformation. The simple function is based on the formula: $Proportion = (\frac{MaxDistance - distance}{MaxDistance})^a$ where a is a scaling constant. On the other hand, the Gaussian function is based on the formula: $Proportion = exp(-(\frac{distance^2}{2 \times SD^2}))$ where $SD = \frac{MaxDistance}{3}$. Since spatial locality is normal in nature [5], the Gaussian function is adopted in CrowdingDE-L for the transformation if not specified explicitly.

Algorithm 1. Trial Vector Generation Using Spatial Locality

P: Parent individual
$trial$: Trial vector
D_G: Genome dimension
CR: Crossover probability
$I[i]$: The gene value of individual I at dimension i
$rand()$: A random number in the interval $[0,1]$

procedure NEWTRIALVECTORGENERATION(P)

1. Transform the distances between P and all candidate individuals to proportions using a transformation function;
2. Prepare a roulette-wheel based on the transformed proportions;
3. Use the roulette-wheel to pick 3 different individuals I_1, I_2, I_3 where $P \neq I_1 \neq I_2 \neq I_3$;

$trial = P$;
$i \leftarrow \lfloor rand() \times D_G \rfloor$;
for ($k = 0$; $k < D_G$; $k = k + 1$) **do**
  if $rand() < CR$ **then**
    $trial[i] = I_1[i] + F \times (I_2[i] - I_3[i])$;
    $i = (i + 1)$ *mod* D_G;
  end if
end for
return $trial$;
end procedure

4 Experiments

We implemented all the algorithms using Sun's Java programming language. The development was based on the EC4 framework [4]. Experiments to compare the performance among CrowdingDE-L and other algorithms were conducted on ten benchmark functions. The other algorithms include: Crowding Genetic Algorithm (CrowdingGA) [3], CrowdingDE [18], Fitness Sharing Genetic Algorithm (SharingGA) [8], SharingDE [18], Species Conserving Genetic Algorithm (SCGA) [9], and SDE [10]. The first five benchmark functions are widely adopted in literatures: F1 is Deb's 1st function [19], F2 is Himmelblau function [1], F3 is Six-hump Camel Back function [12], F4 is Branin function [12] and F5 is Rosenbrock function [16]. The remaining five benchmark functions were derived from [19,11].

4.1 Performance Measurements

For multimodal optimization, there are several performance metrics proposed [11,10,9,18]. The focuses of this paper are on the ability of the algorithms to locate the optima and the accuracy of the optima found by the algorithms. Hence we adopted the Peak Ratio (PR) and Average Minimum Distance to the Real Optima (D) [11,19] as the performance metrics.

As different algorithms perform different operations in one generation, it is unfair to set the termination condition as the number of generations. Alternatively, it is also unfair to adopt CPU time, since it substantially depends on the implementation techniques for different algorithms. For instance, the sorting techniques and the programming languages used. In contrast, fitness function evaluation is always the performance bottleneck. Thus the number of fitness function evaluations was set as the termination condition in the following experiments. All algorithms were run up to a maximum of 40000 fitness function evaluations. The above performance metrics were obtained by taking the average and standard deviation of 50 runs.

4.2 Parameter Setting

Sun's Java Double (double-precision 64-bit IEEE 754 floating point) was used as the real number representation for all algorithms. All populations were initialized randomly. The random seed was 12345. For all DE algorithms, the crossover probability (CR) was 0.9 and F was 0.5. The common GA parameter settings of CrowdingGA, SharingGA and SCGA for all benchmarks were the same as Table 5 in [19]. For all crowding algorithms, population size was set to 100 for Peaks2, Peaks3 and Peaks4. 50 was set for the remaining benchmark functions. The parameter settings of SharingDE, SharingGA, SCGA and SDE for different benchmarks are tabulated in Table 1. For SharingDE and SharingGA, σ and α denote the niche radius and scaling factor respectively. The parameters have been tuned in a few preliminary runs with manual inspections for all algorithms.

Table 1. Parameter setting of SharingDE, SharingGA, SCGA and SDE for different benchmarks

		SharingDE [18]		SharingGA [8]		SCGA [9] and SDE [10]
Benchmark	Population Size	α	σ	α	σ	Species Distance*
F1	100	1	0.001	1	0.001	0.01
F2	100	1	0.03	5	0.1	3
F3	100	1	0.01	2	40	0.5
F4	100	1	0.01	1	0.1	6
F5	100	3	30	3	30	10
Peaks1	200	1	100	1	50	50
Peaks2	200	1	100	2	50	25
Peaks3	200	1	5	1	0.5	3
Peaks4	200	1	5	1	0.5	3
Peaks5	200	1	300	1	300	150

4.3 Results

Table 2 shows the experimental results. Mann-Whitney U tests (two-tailed) have been conducted to reject the null hypothesis that the performance metrics of CrowdingDE-L and those of another algorithm are sampled from the same population. For D in all

Species Distance $= \sigma_s/2$ in [9].

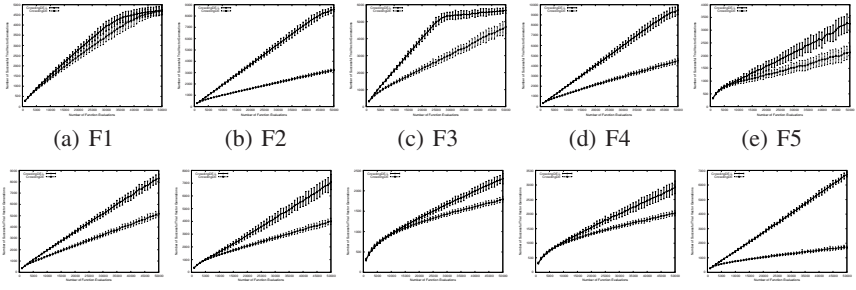
Table 2. Experimental Results for all algorithms tested (averaged over 50 runs)

Benchmark	Measurement	CrowdingDE-L	CrowdingGA [3]	CrowdingDE [18]	SharingGA [8]	SharingDE [18]	SDE [10]	SCGA [9]
	Mean of D	**2.81E-10**	2.24E-06	3.72E-10	4.08E-03	1.14E-03	1.59E-03	1.09E-03
	StDev of D	**8.00E-11**	4.81E-06	2.03E-10	1.21E-02	4.53E-04	7.87E-03	1.16E-03
	Minimum of D	**8.44E-11**	4.35E-09	1.38E-10	2.34E-05	4.66E-04	8.43E-07	6.06E-05
F1	Median of D	**3.31E-11**	2.24E-06	2.82E-10	1.28E-04	1.33E-03	4.15E-06	1.13E-03
	Mean of Peak Ratio	**1.00**	**1.00**	**1.00**	0.98	**1.00**	0.99	**1.00**
	StDev of Peak Ratio	**0.00**	**0.00**	**0.00**	0.06	**0.00**	0.04	**0.00**
	Mean of D	**2.20E-07**	4.93E-04	3.86E-05	2.06E+00	4.92E-01	1.20E+00	2.59E-01
	StDev of D	**1.53E-06**	5.49E-04	1.51E-05	1.05E+00	7.77E-01	6.36E-01	1.17E-01
	Minimum of D	**2.63E-10**	1.30E-05	1.11E-05	7.39E-03	2.63E-02	2.60E-03	2.34E-02
F2	Median of D	**2.68E-09**	3.03E-04	5.49E-05	7.42E-01	5.01E-02	1.23E+00	1.63E-01
	Mean of Peak Ratio	**1.00**	**1.00**	**1.00**	0.66	0.91	0.78	0.44
	StDev of Peak Ratio	**0.00**	**0.00**	**0.00**	0.17	0.14	0.11	0.18
	Mean of D	**1.66E-09**	2.21E-05	4.86E-07	1.44E-01	1.55E-02	6.22E-03	2.03E-02
	StDev of D	**4.35E-10**	3.38E-05	4.58E-07	2.89E-01	4.96E-03	2.26E-03	2.22E-02
	Minimum of D	**6.07E-10**	3.52E-08	9.41E-08	2.14E-04	4.55E-03	1.49E-03	3.54E-04
F3	Median of D	**1.76E-09**	1.49E-06	5.75E-07	5.85E-04	1.63E-02	5.25E-03	5.62E-02
	Mean of Peak Ratio	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	0.95
	StDev of Peak Ratio	**0.00**	**0.00**	**0.00**	0.20	**0.00**	**0.00**	0.15
	Mean of D	**4.87E-07**	5.96E-02	2.45E-04	3.39E+00	1.38E+00	2.61E-01	6.63E-01
	StDev of D	**1.50E-06**	1.14E-01	1.25E-04	1.99E+00	1.85E+00	7.81E-01	5.91E-01
	Minimum of D	**1.20E-08**	3.08E-05	6.44E-05	6.09E-03	7.94E-03	3.84E-03	7.72E-02
F4	Median of D	**1.62E-07**	1.22E-01	3.74E-04	5.84E+00	1.96E+00	1.06E+00	4.22E-01
	Mean of Peak Ratio	**1.00**	0.89	**1.00**	0.61	0.88	0.97	0.41
	StDev of Peak Ratio	**0.00**	0.16	**0.00**	0.21	0.16	0.10	0.14
	Mean of D	**7.81E-03**	1.22E-02	2.28E-02	8.59E-03	4.14E-02	4.23E-02	8.76E-03
	StDev of D	**7.54E-03**	2.84E-02	2.03E-02	1.83E-02	1.40E-01	3.07E-02	1.11E-02
	Minimum of D	3.77E-04	2.31E-05	1.11E-03	1.54E-05	1.63E-09	1.29E-03	**9.38E-13**
F5	Median of D	**5.05E-03**	2.57E-02	1.10E-02	6.84E-03	3.63E-02	5.24E-02	7.55E-03
	Mean of Peak Ratio	**1.00**	0.96	**1.00**	**1.00**	0.92	0.94	**1.00**
	StDev of Peak Ratio	**0.00**	0.20	**0.00**	**0.00**	0.27	0.24	**0.00**
	Mean of D	**4.95E-07**	6.36E-01	7.79E-06	4.84E+00	2.67E+01	5.00E+01	2.11E+00
	StDev of D	**2.12E-06**	1.18E+00	5.58E-06	7.78E+00	3.12E+01	7.71E+00	9.10E-01
	Minimum of D	**1.08E-10**	1.62E-05	1.16E-06	6.88E-01	2.90E-01	3.74E+01	4.80E-01
Peaks1	Median of D	**1.77E-09**	9.40E-04	1.64E-05	1.87E+01	8.00E+00	5.48E+01	1.92E+00
	Mean of Peak Ratio	**1.00**	0.86	**1.00**	0.01	0.37	0.21	0.34
	StDev of Peak Ratio	**0.00**	0.17	**0.00**	0.05	0.10	0.16	0.05
	Mean of D	1.30E+01	7.43E+00	1.24E+01	3.66E+01	3.60E+01	6.92E+01	**5.76E+00**
	StDev of D	**8.64E-01**	3.52E+00	1.51E+00	4.55E+00	1.01E+01	8.08E+00	1.24E+00
	Minimum of D	9.65E+00	**1.49E+00**	9.29E+00	2.10E+01	1.43E+01	6.58E+01	3.29E+00
Peaks2	Median of D	1.33E+01	**4.68E+00**	1.14E+01	3.59E+01	3.38E+01	6.60E+01	6.11E+00
	Mean of Peak Ratio	**0.70**	0.36	0.66	0.00	0.21	0.19	0.11
	StDev of Peak Ratio	0.03	0.07	0.08	**0.02**	0.04	0.04	**0.02**
	Mean of D	**5.75E-03**	9.15E-02	1.44E-02	1.59E+00	2.24E-01	3.69E+00	4.07E-01
	StDev of D	**1.43E-03**	8.21E-02	2.04E-03	3.66E-01	1.28E-01	6.48E-01	1.09E-01
	Minimum of D	**3.32E-03**	1.89E-02	1.05E-02	9.92E-01	7.05E-02	2.34E+00	2.57E-01
Peaks3	Median of D	**5.64E-03**	6.31E-02	1.44E-02	1.92E+00	2.44E-01	4.72E+00	3.77E-01
	Mean of Peak Ratio	**1.00**	0.84	**1.00**	0.61	0.55	0.34	0.27
	StDev of Peak Ratio	**0.00**	0.09	**0.00**	0.07	0.12	0.05	0.08
	Mean of D	**3.24E-03**	2.69E-01	1.76E-02	2.49E+00	4.90E-01	3.36E+00	5.20E-01
	StDev of D	**8.89E-04**	1.75E-01	2.87E-02	4.32E-01	1.81E-01	6.96E-01	1.19E-01
	Minimum of D	**1.89E-03**	5.56E-02	8.77E-03	1.92E+00	1.77E-01	2.47E+00	3.29E-01
Peaks4	Median of D	**3.95E-03**	1.73E-01	1.37E-02	2.90E+00	4.55E-01	3.37E+00	5.31E-01
	Mean of Peak Ratio	**1.00**	0.69	**1.00**	0.24	0.33	0.33	0.20
	StDev of Peak Ratio	**0.00**	0.08	0.01	0.09	0.12	0.04	0.06
	Mean of D	**7.80E-01**	1.29E+02	6.14E+00	1.84E+02	3.19E+01	2.81E+01	1.13E+02
	StDev of D	1.50E+00	1.58E+01	**1.14E+00**	1.54E+01	2.46E+01	2.85E+01	1.70E+01
	Minimum of D	**2.80E-03**	8.71E+01	2.37E+00	1.40E+02	1.04E+01	1.24E-02	8.39E+01
Peaks5	Median of D	**1.06E-01**	1.25E+02	6.33E+00	1.93E+02	3.41E+01	2.57E+01	1.04E+02
	Mean of Peak Ratio	**0.83**	0.00	0.00	0.00	0.00	**0.83**	0.00
	StDev of Peak Ratio	**0.14**	0.00	0.00	0.00	0.00	0.24	0.00

benchmarks, except that with SCGA in F5, their differences are found statistically significant with α-level=0.01. For PR in all benchmarks, except that with SDE in F1, F5, and Peaks5 and that with CrowdingGA in F5, their differences are found statistically significant with α-level=0.05. On the whole, CrowdigDE-L showed its competitive results with other existing algorithms.

4.4 Effect of Spatial Locality

To demonstrate the effect of spatial locality, experiments between CrowdingDE-L and CrowdingDE were conducted on all the benchmark functions. Figure 2 depicts the results.



(a) F1 (b) F2 (c) F3 (d) F4 (e) F5

(f) Peaks1 (g) Peaks2 (h) Peaks3 (i) Peaks4 (j) Peaks5

Fig. 2. Effect of Spatial Locality (averaged over 50 runs)

Each sub-figure corresponds to one benchmark function. The vertical axis is the number of successful trial vector generation (averaged over 50 runs). A successful trial vector generation is defined as the generation of an offspring, which can replace an individual in a parent population. On the other hand, the horizontal axis is the number of fitness function evaluations. It could be observed that CrowdingDE-L generate a higher number of successful trial vectors than CrowdingDE no matter how the fitness function evaluations were varied from 1000 to 50000. More chances for convergence were provided in CrowdingDE-L.

5 Real World Application

To verify the proposed algorithm in real world optimization, we have adopted an optical system design problem [14,13] as a benchmark problem.

5.1 Varied-Line-Spacing Holographic Grating Design

Holographic gratings have been widely applied in optical instruments for aberration corrections. In particular, the Varied-Line-Spacing (VLS) holographic grating distinguishes itself from others by the high order aberration eliminating capability in diffractive optical systems. It is commonly used in high resolution spectrometers and

monochromaters. A recording optical system of VLS holographic grating is outlined in [14].

The objective for the design is to find several sets of design variables (or recording parameters [14]) to form the expected groove shape of G (or the distribution of groove density [13]). Mathematically, the goal is to minimize the definite integral of the square error between the expected groove density and practical groove density [14]:

$$min \;\; J = \int_{-w_0}^{w_0} (n_p - n_e)^2 dw$$

where w_0 is the half-width of the grating, n_p is the practical groove density and n_e is the expected groove density. These two groove densities are complicated functions of the design variables [14].

Theoretically, the above objective is simple and clear. Unfortunately, in practice, there are many other auxiliary optical components, which constraints are too difficult to be expressed and solved in mathematical forms. Single optimal solution is not necessarily a feasible and favorable solution. Thus optical engineers often need to tune the design variables to find as many optimal solutions as possible for multiple trials. Multimodal optimization becomes necessary for the design problem.

5.2 Performance Measurements

As the objective function is an unknown landscape, the exactly optimal information is not available. Thus the previous performance metrics cannot be simply adopted in this section. We propose two new performance metrics in this section. The first one is the best fitness, which is the fitness value of the fittest individual in the last population. The second one is the number of distinct peaks, where a distinct peak is considered found when there exists an individual which fitness value is below a threshold 0.0001 and there isn't an individual within 0.1 distance unit and found as a peak before in the last population. The threshold is chosen to 0.0001 because the fitness values of the solutions found in [14] is around this order of magnitude. On the other hand, the distance is chosen to 0.1 unit because it has already been set for considering peaks found in peak ratio [19,11].

5.3 Parameter Setting

Same as before, all algorithms were run up to a maximum of 40000 fitness function evaluations. The above performance metrics were obtained by taking the average and standard deviation of 50 runs. The groove density parameters followed the setting in [14]: $n_0 = 1.400 \times 10^3 (line/mm)$, $b_2 = 8.2453 \times 10^{-4} (1/mm)$, $b_3 = 3.0015 \times 10^{-7} (1/mm^2)$ and $b_4 = 0.0000 \times 10^{-10} (1/mm^3)$. Half-width w_0 was 90mm. The radii of spherical mirrors M_1 and M_2 were 1000mm. The recording wavelength (λ_0) was 413.1nm. The previous settings were adopted except the algorithm-specific parameters: The species distance of SDE and SCGA was set to 500. The scaling factor and niche radius of SharingDE and SharingGA were set to 1 and 1000 respectively. The population size was set to 50.

5.4 Results

The result is tabulated in Table 3. CrowdingDE-L showed slightly better results among the algorithms.

Table 3. Experimental Results for all algorithms tested on the VLS holographic grating design

Measurement	CrowdingDE-L	CrowdingGA [3]	CrowdingDE [18]	SharingGA [8]	SharingDE [18]	SDE [10]	SCGA [9]
Mean of Best Fitness	**4.47E-10**	1.94E-09	2.88E-07	5.05E-03	2.41E+02	1.38E-01	7.81E+03
StDev of Best Fitness	**3.16E-09**	1.06E-08	2.04E-06	1.67E-02	3.45E+02	2.34E-01	1.52E+04
Min of Best Fitness	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.18E-02	1.69E-04	9.35E-01
Median of Best Fitness	1.12E-08	**0.00E+00**	**0.00E+00**	5.33E-06	4.08E+01	7.09E-02	8.46E+03
Means of Peaks Found	**50.00**	17.08	**50.00**	2.16	0.00	0.00	0.00
StDev of Peaks Found	**0.00**	4.55	**0.00**	3.47	0.00	0.00	0.00
Min of Peaks Found	**50.00**	10.00	**50.00**	0.00	0.00	0.00	0.00
Median of Peaks Found	**50.00**	15.50	**50.00**	1.50	0.00	0.00	0.00

6 Conclusion

Confirmed by the experimental results, CrowdingDE-L is highlighted with its ability for generating fitter trial vectors, which can successfully replace parent individuals. Extensive experiments have been conducted. The results indicate that CrowdingDE-L has its own competitive edge over the other algorithms tested, in terms of the performance metrics.

The locality principle is proven simple and useful in computing [5]. In a macro-view, the work in this paper can be regarded as a case study for integrating the locality principle into an evolutionary algorithm. The numerical results can also be viewed as a valuable resource for comparing the state-of-the-art algorithms for multimodal optimization.

Acknowledgment

The authors are grateful to anonymous reviewers for their valuable comments. The authors would also like to thank Ling Qing for his source codes and insightful discussions. This research is partially supported by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. 414107 and 414708).

References

1. Beasley, D., Bull, D.R., Martin, R.R.: A sequential niche technique for multimodal function optimization. Evol. Comput. 1(2), 101–125 (1993)
2. Bersini, H., Dorigo, M., Langerman, S., Seront, G., Gambardella, L.: Results of the first international contest on evolutionary optimisation (1st ICEO). In: Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, May 1996, pp. 611–615 (1996)
3. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan, Ann Arbor (1975); University Microfilms No. 76-9381
4. De Jong, K.A.: Evolutionary Computation. A Unified Approach. MIT Press, Cambridge (2006)
5. Denning, P.J.: The locality principle. Commun. ACM 48(7), 19–24 (2005)

6. Feoktistov, V.: Differential Evolution - In Search of Solutions. Springer Optimization and Its Applications, vol. 5. Springer-Verlag New York, Inc., Secaucus (2006)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
8. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms and their application, pp. 41–49. L. Erlbaum Associates Inc., Hillsdale (1987)
9. Li, J.P., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. 10(3), 207–234 (2002)
10. Li, X.: Efficient differential evolution using speciation for multimodal function optimization. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 873–880. ACM, New York (2005)
11. Lung, R.I., Chira, C., Dumitrescu, D.: An agent-based collaborative evolutionary model for multimodal optimization. In: GECCO 2008: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation, pp. 1969–1976. ACM, New York (2008)
12. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, London (1996)
13. Qing, L., Gang, W., Qiuping, W.: Restricted evolution based multimodal function optimization in holographic grating design. In: The 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, September 2005, vol. 1, pp. 789–794 (2005)
14. Qing, L., Gang, W., Zaiyue, Y., Qiuping, W.: Crowding clustering genetic algorithm for multimodal function optimization. Appl. Soft Comput. 8(1), 88–95 (2008)
15. Rogers, A., Pingali, K.: Process decomposition through locality of reference. SIGPLAN Not. 24(7), 69–80 (1989)
16. Shang, Y.W., Qiu, Y.H.: A note on the extended rosenbrock function. Evol. Comput. 14(1), 119–126 (2006)
17. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997), http://www.springerlink.com/content/x555692233083677/
18. Thomsen, R.: Multimodal optimization using crowding-based differential evolution. In: Congress on Evolutionary Computation, CEC 2004, June 2004, vol. 2, pp. 1382–1389 (2004)
19. Wong, K.C., Leung, K.S., Wong, M.H.: An evolutionary algorithm with species-specific explosion for multimodal optimization. In: GECCO 2009: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 923–930. ACM, New York (2009)

A Directed Mutation Operator for Real Coded Genetic Algorithms

Imtiaz Korejo, Shengxiang Yang, and Changhe Li

Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, UK
{iak5,s.yang,cl160}@mcs.le.ac.uk

Abstract. Developing directed mutation methods has been an interesting research topic to improve the performance of genetic algorithms (GAs) for function optimization. This paper introduces a directed mutation (DM) operator for GAs to explore promising areas in the search space. In this DM method, the statistics information regarding the fitness and distribution of individuals over intervals of each dimension is calculated according to the current population and is used to guide the mutation of an individual toward the neighboring interval that has the best statistics result in each dimension. Experiments are carried out to compare the proposed DM technique with an existing directed variation on a set of benchmark test problems. The experimental results show that the proposed DM operator achieves a better performance than the directed variation on most test problems.

1 Introduction

Genetic algorithms (GAs) are a class of probabilistic optimization techniques inspired by genetic inheritance and natural evolution. GAs have been used for solving many optimization problems due to the properties of self-learning, self-organization, and self-adaptation, as well as the properties of implicit parallelism [1,5]. GAs are population based approaches, where new populations are generated by the iterative application of selection and variation of individuals in the population. For example, mutation is used to explore new solutions, crossover exchanges genetic information between two individuals, and selection selects relatively fit individuals for the next population. The fitness of an individual is evaluated by a fitness function, which defines the external environment of a GA. The performance of a GA depends on not only the above variation operators, but also some other factors, e.g., the population size and selection method, etc.

Mutation is a key operator to increase the diversity of the population and hence enables GAs to explore promising areas of the search space [9]. The step size and search direction are major factors that determine the performance of mutation operators. It may be beneficial to use different values during different stages of evolution in order to get a better performance of GAs. However, it is impossible to know the optimum mutation step size and search direction

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 491–500, 2010.

for real-world problems. Hence, it is usually beneficial to use some statistics information to guide the mutation operator for GAs [4]. Strategy parameters are adjusted according to one of the three methods: deterministic adaptation adjusts the values of parameters according to predefined rules without using any learning information from GAs; adaptive adaptation alters the parameters using some learning information from the search space. The best example of adaptive adaptation is Rechenberg's '1/5' success rule in evolutionary strategies; and self-adaptive adaptation embeds the parameters into the chromosomes of individuals and modifies the parameters by the GA itself.

Several researchers have tried to increase the performance of real-coded GAs by using directed mutation techniques [2,6,11]. In [3], the authors proposed a co-evolutionary technique where each component of a solution vector is added one extra bit to determine the direction of mutation. The direction bit is adapted by using the feedback information from the current population. A directed mutation based on momentum was proposed in [11], where each component of an individual is attached a standard gaussian mutation and the current momentum to mutate that component.

In this paper, a new directed mutation technique is proposed for GAs to explore promising areas in the search space. This approach first calculates the statistical information from the current population regarding the average fitness and the number of individuals distributed within each interval of each dimension of the search space. Then, the statistical information is used to guide the mutation of an individual toward the neighboring interval that has a better statistics result in each dimension. In order to investigate the performance of the proposed directed mutation, an experimental study is carried out to compare the performance of the GA with the proposed directed mutation and the GA with the directed variation in [13].

2 Related Work

A directed variation (DV) technique was proposed by Zhou and Li [13]. This algorithm does not introduce a scheme for completely generating an average step size but adjusts some individuals by using the feedback information from the current population. Suppose the population is a set of N individuals $X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, \cdots, \overrightarrow{x}_N\}$ and each individual is a K-dimensional vector, denoted by $\overrightarrow{x}_i = [x_{i1}, x_{i2}, \cdots, x_{iK}]$. Denote the minimal d-th component of the individuals at generation t by x_d^L and the maximum by x_d^U, that is, the range of the d-th dimension at time t is $R_d(t) = [x_d^L, x_d^U]$. This range can be equally divided into L intervals. The fitness of a nonempty interval, say, the j-th interval of the d-th dimension, is defined by:

$$F_{dj} = \sum_{i=1}^{N} I(x_{id} \in B_{dj}) f_{Norm}(\overrightarrow{x}_i) \tag{1}$$

$$I(x_{id} \in B_{dj}) = \begin{cases} 1, \text{ if } x_{id} \in B_{dj} \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

where B_{dj} denotes the range (lower and upper bounds) of the j-th interval of the d-th dimension, N presents the population size, $I(.)$ is the indicator function, and the fitness of each solution vector $\overrightarrow{x}_i$ is normalized as follows:

$$f_{Norm}(\overrightarrow{x}_i) = \frac{f(\overrightarrow{x}_i) - f_{min}}{f_{max} - f_{min}} \tag{3}$$

where f_{max} and f_{min} represent the maximum and minimum fitness of the whole population respectively.

With DV, in each generation some individuals are selected for directed variation in each component. DV is applied on a component, say, the d-th component, only when the range of the d-th dimension of all solutions in the current generation decreases in comparison with that of the previous generation, i.e., the population converges regarding that dimension. DV works as follows. First, the fitness of interval, i.e., F_{dj}, is calculated according to Eq. (1). Then, DV is applied for an individual component by component, where each component of the individual may be shifted from its current interval to a neighboring interval that has a higher fitness with a certain probability, as described below.

In DV, for each component $x_{id} \in B_{dj}$ of an individual $\overrightarrow{x}_i$, whether it is mutated depends on the value F_{dj} and the fitness of its neighboring intervals, i.e., $F_{d,j-1}$ and $F_{d,j+1}$. If F_{dj} is bigger than both $F_{d,j-1}$ and $F_{d,j+1}$, then DV is not applied to the d-th component of any selected individuals with $(x_{id}) \in B_{dj}$. If F_{dj} is in the middle, without loss of generality, suppose $F_{d,j-1} > F_{dj} > F_{d,j+1}$, the probability of directed variation, P_{dj}^{DV}, can be calculated as follows:

$$P_{dj}^{DV} = 1 - \frac{F_{dj}}{F_{d,j-1}} \tag{4}$$

With this probability, x_{id} is replaced with a number, randomly generated between x_{id} and the center of $B_{d,j-1}$. If F_{dj} is smaller than both $F_{d,j-1}$ and $F_{d,j+1}$, then either $B_{d,j-1}$ or $B_{d,j+1}$ is randomly selected with an equal probability and x_{id} moves towards the selected interval, i.e., replaced with a number randomly generated between x_{id} and the center of the selected interval.

3 Directed Mutation for Genetic Algorithms

The main motivation behind directed mutation (DM) is to explore promising areas in the search space by using the feedback information from the current population, e.g., the fitness and some other factors. It is a modified version of the standard mutation. Since individuals in DV only move toward the interval with the highest fitness, it may easily cause the premature convergence problem. This paper introduces a DM technique which aims to explore promising areas of the search space with fixed boundaries according to the fitness of intervals and the percentage of individuals in each interval of each dimension.

In the proposed DM operator, individual shifting is not only based on the feedback information of the average fitness of intervals, but also on the population

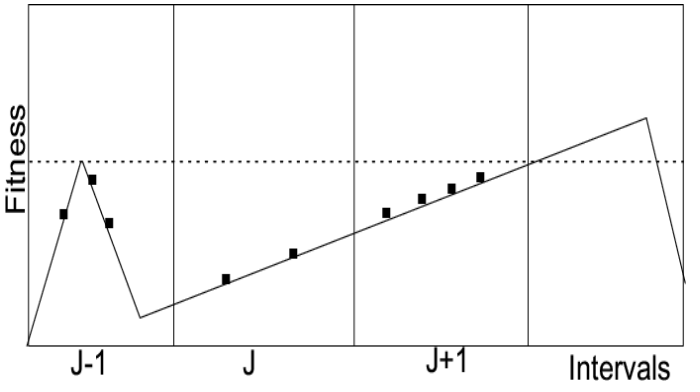


Fig. 1. Fitness landscape of the d-th dimension

distribution. By taking into account the information of population distribution, DM efficiently avoids the premature convergence problem. The key idea of the DM operator is illustrated in Fig. 1.

From Fig. 1, we consider the j-th interval, if we only consider DV, the two individuals of interval j will move toward the $(j-1)$-th interval due to the higher fitness of the $(j-1)$-th interval. However, the right direction should be the $(j+1)$-th interval since the $(j+1)$-th interval is more promising than the $(j-1)$-th interval. Hence, DM is an enhanced version of DV.

The framework of the GA with the proposed DM operator is given in Algorithm 1. The proposed GA differs from the standard GA in that in each generation, a set of individuals are selected to undergo the DM operation iteratively. As shown in Algorithm 2, the DM operator is applied for each component of a selected solution in a similar way as the DV operator described above. The difference lies in the calculation of the probability of moving a component of a solution from one interval to its neighboring interval, which is described in detail below.

Similar to the DV operator, the range of the d-th dimension of individuals at generation t, i.e., $R_d(t) = [x_d^L, x_d^U]$, is also equally divided into L intervals. DM is applied only when the range of the d-th component $R_d(t)$ of all solution vectors of current generation t decreases in comparison with that of previous generation $t-1$. The fitness F_{dj} of each non-empty interval is calculated by Eq. (1), Eq. (2), and Eq. (3). In addition to the fitness of each interval, the percentage of individuals in each interval is also calculated in the DM operator as follows:

$$P_{dj} = \frac{1}{N} \sum_{i=1}^{N} I(x_{id} \in B_{dj}) \tag{5}$$

where P_{dj} represents the percentage of individuals with the d-th component in the j-th interval in the current population, and B_{dj} and $I(.)$ are the same as defined before in Eq. (1) and Eq. (2). From F_{dj} and P_{dj}, we calculate a

Algorithm 1. GA with Directed Mutation (DM)

```
Randomly generate an initial population pop
Evaluate the fitness of each individual of pop
t := 0.
while t < max_gen do
    for each individual i in pop do
        Select individual j by the roulette wheel method
        Crossover individual i with individual j using the arithmetic crossover method
        Mutate individual i by using the Gaussian mutation with mean zero and pre-
        selected or adaptive standard deviation
    end for
    Apply DM on a set of individuals randomly selected from the population
    t := t + 1
end while
```

Algorithm 2. Directed Mutation

```
for each dimension d ∈ {1, 2, ··· , K} do
    if R_d(t) < R_d(t − 1) then
        for each interval j do
            Calculate F_dj according to Eq. (1)
            Calculate the number of individuals P_dj according to Eq. (5)
        end for
        for each interval j do
            Calculate FP_dj according to Eq. (6)
        end for
        for each interval j do
            Calculate P_dj^DM according to Eq. (7)
        end for
        Shift the genes x_id of selected individuals to their neighboring interval with a
        higher fitness with the associated probability
    end if
end for
```

value that is associated with the j-th interval of the d-th dimension, assuming $F_{d,j-1} > F_{dj} > F_{d,j+1}$, as follows:

$$FP_{dj} = \frac{F_{dj}}{F_{d,j-1}} + \frac{P_{dj}}{P_{d,j-1}} \tag{6}$$

With above definitions, the component x_{id} of a selected individual $\overrightarrow{x}_i$ is mutated by associated value of FP_{dj}. Where F_{dj} is bigger than the fitness of both neighboring intervals, i.e., $F_{d,j-1}$ and $F_{d,j+1}$, no directed mutation will be used to x_{id} . If FP_{dj} is in the middle in comparison with the fitness of its two neighbor intervals $j-1$ and $j+1$, without loss of generality, suppose $FP_{d,j-1} > FP_{dj} > FP_{d,j+1}$. Then, move the individual $\overrightarrow{x}_i$ towards the interval $j-1$ with a certain probability, which is calculated as follows.

$$P_{dj}^{DM} = \frac{FP_{dj}}{\sum_{j=1}^{L} FP_{dj}} \tag{7}$$

where the DM probabilities P_{dj}^{DM} are normalized over all intervals. In this case, the solution $\overrightarrow{x}_i$ is moved toward the interval $j-1$ by replacing x_{id} with a number randomly generated between x_{id} and the center of $B_{d,j-1}$ as follows:

$$x_{id} = rand(x_{id}, B_{d,j-1}) \tag{8}$$

Otherwise, if $FP_{dj} < FP_{d,j-1}$ and $FP_{dj} < FP_{d,j+1}$, then either $B_{d,j-1}$ or $B_{d,j+1}$ is selected with an equal probability and the solution $\overrightarrow{x}_i$ moves towards the selected interval with the probability P_{dj}^{DM}.

4 Experimental Study

4.1 Experimental Setting

In order to test the performance of GA with DM, three unimodal functions and eleven multimodal functions, which are widely used as the test functions in the literature [10,12], were selected as the test bed in this paper. The number of dimensions n is set to 10 for all test functions. The details of these test functions are given in Table 1. Function f_9 is a composition function proposed by Suganthan et al. [10], which is composed of ten benchmark functions: the rotated version and shifted version of f_1, f_2, f_3, f_4, and f_5, as also listed in Table 1, respectively. Functions f_{10} to f_{14} are rotated functions, where the rotation matrix $\boldsymbol{M}$ for each function is obtained using the method in [7].

The idea of DV was taken from [13], which is implemented in the peer GA. The adaptive standard deviation [8] is used in DM. The population size (100) and total number of generations (500) are the same for both DM and DV on all problems, the total number of intervals L was set to 3, 6, 9, and 12, respectively. The mutation probability P_m for the Gaussian mutation is the same for DM and DV, which was set to different values for different test problems, as listed in Table 1. Both the GA with DM and the GA with DV were run 30 times independently on each test problem.

4.2 Experimental Results and Analysis

The average results of 30 independent runs of the GA with directed mutation and the GA with directed variation on the test problems are shown in Table 2. From Table 2, it can be seen that the number of intervals used for each dimension is a key parameter. The performance of the GA with DM becomes significantly better on some problems than that of the GA with DV as the number of intervals increases. The performance of both operators is different on different problems.

When the number of intervals is set to 3, the results of DM are better than DV on half of the test problems. DM is trapped into local optima due to the

Table 1. Test functions of $n = 10$ dimensions, where D ($D \in R_n$) and f_{min}, denote the domain and the minimum value of a function respectively. $\boldsymbol{M}$ is rotation matrix.

Test Function	P_m	D	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	0.1	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	0.01	[-5.12,5.12]	0
$f_3(x) = \sum_{i=1}^{n} \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))] - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$, $a = 0.5, b = 3, k_{max} = 20$	0.01	[-0.5,0.5]	0
$f_4(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	0.01	[-600, 600]	0
$f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	0.01	[-32, 32]	0
$f_6(x) = \sum_{i=1}^{n} 100(x_{i+1}^2 - x_i)^2 + (x_i - 1)^2)$	0.05	[-30, 30]	0
$f_7(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	0.01	[-500, 500]	-4189.829
$f_8(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	0.01	[-100, 100]	0
$f_9(x) =$ Composition function (CF 5) in [10]	0.05	[-5, 5]	0
$f_{10}(x) = \sum_{i=1}^{n} 100(y_{i+1}^2 - y_i)^2 + (y_i - 1)^2), \boldsymbol{y} = \boldsymbol{M} * \boldsymbol{x}$	0.05	[-100, 100]	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} (y_i - 100)^2 - \prod_{i=1}^{n} \cos(\frac{y_i - 100}{\sqrt{i}}) + 1$, $\boldsymbol{y} = \boldsymbol{M} * \boldsymbol{x}$	0.01	[-600, 600]	0
$f_{12}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi y_i)) + 20 + \text{e}$, $\boldsymbol{y} = \boldsymbol{M} * \boldsymbol{x}$	0.01	[-32,32]	0
$f_{13}(x) = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10), \boldsymbol{y} = \boldsymbol{M} * \boldsymbol{x}$	0.01	[-5, 5]	0
$f_{14}(x) = \sum_{i=1}^{n} \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (y_i + 0.5))] - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$, $a = 0.5, b = 3, k_{max} = 20, \boldsymbol{y} = \boldsymbol{M} * \boldsymbol{x}$	0.01	[-0.5,0.5]	0

large range of intervals. It is interesting that DM achieves the best result on f_9, f_{10}, and f_{12} over all different number of intervals.

When we increase the number of intervals to 6, the performance of DM is better than DV on most benchmark problems. Although DV presents better results than DM on f_1, f_2, f_9, and f_{14}, DM obtains close results to DV on these functions.

Similar observations can be made as the number of intervals increases to 9. The results obtained by DM are better than that of DV. Compared with the results of DM with the number of intervals of 6, the performance of DM deteriorates on some multimodal problems. However, the results of DM with the number of intervals of 9 are better than the results with the number of intervals of 6 on most unimodal problems.

When $L = 12$, the results of DM are better than the results of DV on most test functions. Similar results can be viewed as the number of interval 6 but the performance of DM increases compared with the number of interval of 9 on some multimodal problems.

Table 2. Comparison results between DV and DM with the number of intervals for each dimension set to different values for different problems

function		f_1	f_2	f_3	f_4	f_5	f_6	f_7
$L = 3$	DV	8.28e-06	0.0151	0.2590	0.0985	0.0088	49.41	-2667
	DM	7.33e-06	0.0122	0.2304	0.1116	0.0082	33.87	-2062
$L = 6$	DV	7.55e-06	0.0093	0.2030	0.0820	0.0114	95.25	-2692
	DM	8.99e-06	0.0172	0.1837	0.0385	0.0090	18.84	-1797
$L = 9$	DV	6.19e-06	0.0087	0.2181	0.0731	0.0076	68.78	-2574
	DM	9.03e-06	0.0067	0.2558	0.1143	0.0089	22.97	-1857
$L = 12$	DV	7.21e-06	0.0191	0.2271	0.0972	0.0107	16.63	-2578
	DM	7.86e-06	0.0153	0.2196	0.0341	0.0080	5.65	-1905
t-test	DV-DM	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$+$	-
function		f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
$L = 3$	DV	2.7677	102	61	0.0929	0.0476	2.0971	0.4200
	DM	3.5671	100	21	0.0891	0.0093	2.8655	0.4794
$L = 6$	DV	2.6971	64	82	0.1147	0.0470	2.5969	0.4283
	DM	1.9957	18	14	0.0386	0.0089	2.1034	0.3483
$L = 9$	DV	3.8969	94	30	0.0726	0.0974	2.1034	0.4099
	DM	3.8160	24	24	0.0998	0.0077	2.7950	0.4833
$L = 12$	DV	2.0960	230	50	0.0826	0.0472	3.0315	0.4175
	DM	2.7178	63	53	0.0444	0.0086	1.9218	0.4180
t-test	DV-DM	$\sim$	$+$	$+$	$\sim$	$+$	$\sim$	$\sim$

Table 2 also shows the statistical analysis of comparing DM with DV when $L = 6$ by using the two-tailed t-test with a 58 degree of freedom at a 0.05 level of significance, where the t-test result is presented as "+", "−", or "∼" if the performance of the GA with DM is significantly better than, significantly worse than, or statistically equivalent to the GA with DV, respectively. The DM operator is significantly better on four problems, significantly worse on one problem, and statistically similar on the rest of the problems.

From Table 2, three conclusions can be made. First, the overall performance of DM is better than DV on half test functions at least. Especially, on f_9, f_{10} and f_{12}, the performance of DM is better over all different settings of the number of intervals. Second, the interval quantity is a crucial factor to the performance of both DM and DV on different benchmark functions. According to variable number of intervals, the result of DM and DV varies on different test problems. Third, a larger number of intervals is needed on multimodal problems than unimodal problems. The smaller number of intervals causes the larger number of local optima within an interval, since multimodal problems have many local optima.

Fig. 2 presents the evolutionary process for DM and DV operators on f_4, f_6, f_7, and f_9 respectively, where the result on f_4 is presented in a log scale. From Fig. 2, it can be seen that the convergence speed of DM is faster than that of DV except on f_9. This result validates our idea that the performance of DV can be enhanced by taking into account the population distribution in the search space.

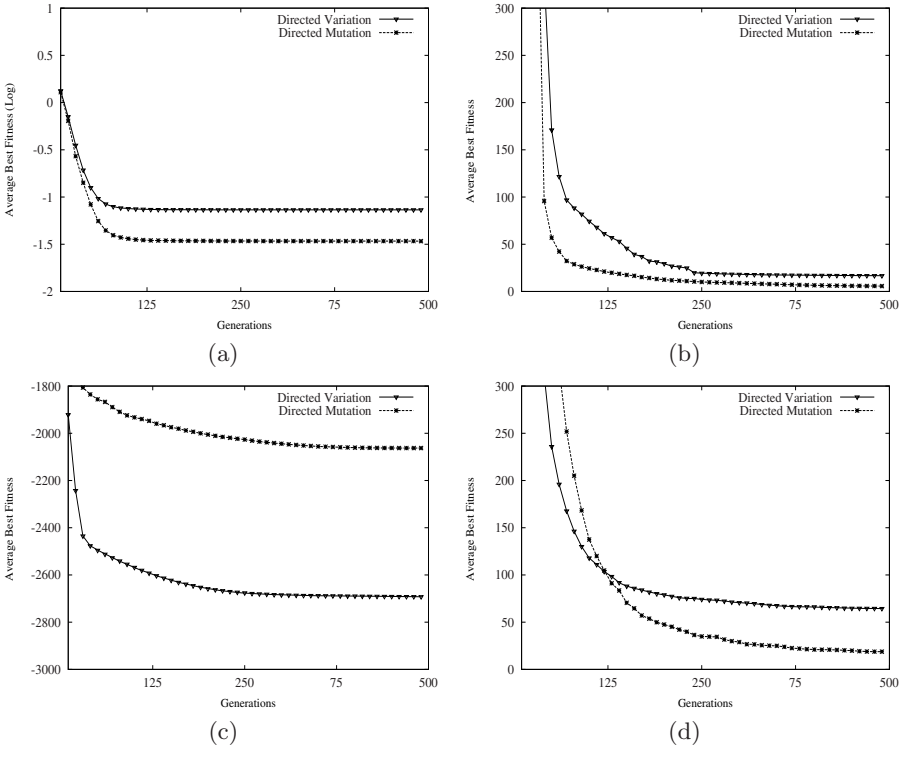


Fig. 2. Evolutionary progress of directed mutation and directed variation operators on (a) f_4 with $L = 12$, (b) f_6 with $L = 12$, (c) f_7 with $L = 3$, and (d) f_9 with $L = 6$

5 Conclusions

In this paper, a directed mutation operator is proposed for genetic algorithms to explore promising solutions in the search space. In the proposed directed mutation, individual shifting is not only based on the feedback information of the fitness of each interval, but also on the population distribution. By taking into account the information of the population distribution, directed mutation greatly improves the performance of directed variation.

In order to justify the proposed directed mutation, a set of benchmark functions was used as the test base to compare the performance of the directed mutation operator with the directed variation operator from the literature [13]. The experimental results show that the efficiency of DV is improved by the proposed enhancement on four out of fourteen functions.

Although DM achieves better results on the test problems, there is a limitation, i.e., different problems need different optimum values of the number of intervals to achieve the best result. So, how to adaptively adjust the number of intervals is our major work in the future, and we will consider to compare

the performance of proposed DM technique with DV on CMA-ES benchmark problems.

References

1. Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford (1996)
2. Berlik, S.: A directed mutation framework for evolutionary algorithms. In: Proc. of the Int. Conf. on Soft Computing, MENDEL, pp. 45–50 (2004)
3. Berry, A., Vamplew, P.: PoD Can Mutate: A simple dynamic directed mutation approach for genetic algorithms. In: Proc. of AISAT 2004: Int. Conf. on Artificial Intelligence in Science and Technology, pp. 200–205 (2004)
4. Eiben, A.E., Michalewics, Z., Schoenauer, M., Smith, J.E.: Parameter control in evolutionary algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithms, ch. 2, pp. 19–46 (2007)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, New York (1989)
6. Hedar, A.R., Fukushima, M.: Directed evolutionary programming: Towards an improved performance of evolutionary programming. In: Proc. of the 2006 IEEE Congress on Evol. Comput., pp. 1521–1528 (2006)
7. Salomon, R.: Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms. BioSystems 39(3), 263–278 (1996)
8. Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley, New York (1995)
9. Spears, W.M.: Crossover or mutation? In: Whitley, L.D. (ed.) Foundations of Genetic Algorithms 2, pp. 81–89. Morgan Kaufmann Publishers, San Mateo (1993)
10. 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 (2005)
11. Temby, L., Vamplew, P., Berry, A.: Accelerating real valued genetic algorithms using mutation-with-momentum. In: Zhang, S., Jarvis, R.A. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 1108–1111. Springer, Heidelberg (2005)
12. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. on Evol. Comput. 3(2), 82–102 (1999)
13. Zhou, Q., Li, Y.: Directed variation in evolutionary strategies. IEEE Trans. on Evol. Comput. 7(4), 356–366 (2003)

Speedups between ×70 and ×120 for a Generic Local Search (Memetic) Algorithm on a Single GPGPU Chip

Frédéric Krüger[1], Ogier Maitre[1], Santiago Jiménez[2], Laurent Baumes[2], and Pierre Collet[1]

[1] LSIIT – UMR 7005, Pôle API, Bd Sébastien Brant, 67400 Illkirch, France
[2] Instituto de Tecnologia Quimica, UPV-CSIC Valencia, Spain
{frederic.kruger,ogier.maitre,pierre.collet}@unistra.fr,
{baumesl,sanjiser}@itq.upv.es

Abstract. This paper presents the first implementation of a generic memetic algorithm on one of the two GPU (Graphic Processing Unit) chips of a GTX295 gaming card. Observed speedups range between ×70 and ×120, mainly depending on the population size.

An automatic parallelization of a memetic algorithm is provided through an upgrade of the EASEA language, so that the EC community can benefit from the extraordinary power of these cards without needing to program them.

1 Introduction

GPGPU (General Purpose Graphic Processing Unit) cards are about to revolutionize evolutionary computation because of their monstrous computing power that evolutionary algorithms can directly use.

Up to now, the best speedups obtained on one GPU chip (*vs* one core of a recent CPU) range between ≈ ×100 on simple EAs (for the complete algorithm) [6] to ≈ ×250 on the more demanding evaluation of a population of diverse GP trees for as few as 32 fitness cases [8].

In the case of memetic algorithms, some papers have been written on the subject, but always in specific cases like Munawar *et al.*, who recently report a ×25 speedup on an expensive TESLA machine thanks to the addition of a local search routine to an evolutionary algorithm to solve the Max-Sat problem [7], or Wong *et al.* who report a maximum speedup of ×4.24 in what may be one of the first papers on the subject [10], or Luo *et al.* on a similar topic [5], but no work has been found that addresses the problem in a non specific (generic) way.

So the aim of this paper is to study the implementation of a *generic* memetic algorithm (*i.e.* an evolutionary algorithm coupled with a local search [3]) on a GPGPU card and the resulting speedup, independently of the problem, the local search, or even the type of evolutionary algorithm.

Then, in order to allow other researchers to use these very specialised cards without needing to understand how to program them, the feature is incorportated

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 501–511, 2010.

in the old EASEA language [1] that creates a complete C++ source file out of the specification of an EA.

2 What Are GPGPUs and How Have They Been Designed?

Some years ago, transistor density on silicon and clock speed seemed so closely related that many people thought that Moore's law applied to the "speed" of computers. This is not true anymore, since even though the doubling of transistors per square millimeters every other year still seems to apply, increase in CPU clock speed has stopped to around 3GHz for many years now.

However Moore's law still applies, meaning that manufacturers still manage to put twice as many transistors on the same surface of silicon every two years.

As a result, manufacturers now increase the power of their chips by making them parallel, which can be done in several ways: one possibility is to more or less duplicate a complete CPU if enough space is available, which is the way chosen by AMD or Intel for their dual or quad-core CPUs. The advantage is that all the different cores can execute independent tasks in parallel (Multiple Instruction Multiple Data paradigm), the price to pay being that it is not possible to put many cores on the chip.

Another possibility is to try to maximise computing power by putting in common all functional units, and use all the available space on the chip to cram in as many Arithmetic and Logic Units as possible. The advantage is that the numbers crunching capacity of the chip becomes monstrous, but the price to pay is that ALUs that have common functional units must execute the same instruction at the same time (Single Instruction Multiple Data paradigm).

2.1 GPU Chips and the Gaming Industry

In order to obtain fluid games with realistic rendering, it is necessary to run the very same algorithm on millions of pixels / vertices as fast as possible. The gaming industry with its billion dollars market (9 billion euros in France only!) therefore elected to create specialised chips that would maximise the number of ALUs, to the detriment of versatility.

nVidia cards such as the 295GTX offer 1.8 TeraFlops for around $400 but with many constraints. These cards host 2×240 cores that are not independent, and with no memory cache (that uses a lot of space on a chip).

Cache memory is not really needed for rendering algorithms because there are many more pixels/vertices than there are available cores. This means that it is possible to stack up many threads on one core, and swap between them whenever one thread does a memory access. This results in some kind of pipelining process that allows to keep the cores busy even with a huge memory latency (600 cycles).

2.2 GPU Chips and Evolutionary Computation

What looks like problematic constraints for most algorithms is fine for Evolutionary Algorithms, as their flowchart is very similar to that of rendering

algorithms: in an evolutionary algorithm, a large population of different genomes (read pixels) needs to be evaluated using the same fitness function (read rendering algorithm). This similarity between the two kinds of algorithms makes it possible for EAs to use GPGPU cards as if they had been designed for EC.

3 Programming GPGPU Cards

The CUDA environment used in this paper allows direct access to nVidia hardware without going through DirectX or OpenGL layers. It is therefore very efficient, but quite difficult to program and portability is limited to nVidia cards.

3.1 Software Architecture

CUDA (released in 2004 by nVidia) is an abstract representation of a unified "computing accelerator." Its goal is to achieve a hardware abstraction of the underlying computing device by defining portable concepts across different architectures without any modification.

The model handles threads of parallel tasks which can be executed on the accelerator. A thread is, as per the classical definition, an independent process, which executes instructions on data. Threads can be arranged in 2D arrays (called blocks), allowing them to map a 2D data array as a 2D texture. The CUDA software architecture defines a complex memory hierarchy, that contains a cache-less large global memory (2×896MB DRAM on a GTX295), read-only cache memories for textures, constants and instructions and small banks of ultra-fast shared memory (16KB) that allow all threads to communicate inside a block.

4 Running a Memetic Algorithm on GPGPU

Memetic algorithms are also referred to as *hybrid algorithms*. They couple a population-based global search (e.g. an evolutionary algorithm) with a local search algorithm, that rapidly optimizes children in a possibly deterministic way.

The algorithm is very similar to a standard evolutionary algorithm: a memetic algorithm creates and initializes a (possibly random) population of individuals after which all individuals are evaluated in order to create a population of parents. Then, until a stopping condition is met, a population of children is created using stochastic genetic operators (such as crossover and mutation) and the point that makes memetic algorithms different from standard evolutionary algorithms is that before the replacement operator elects the individuals that will constitute the next generation, the children undergo a local search optimization.

Here, not only can the evaluation function be computed on the GPU card, but the local search too can be performed on the parallel card. The only problem is that if the local search improves a child, it is then necessary to transfer the new genotype back onto the CPU that runs the evolutionary algorithm.

5 Implementation of a Parallelized Memetic Algorithm

The automatic parallel implementation of a memetic algorithm for the CUDA environment has been incorporated into the EASEA language, that was designed to basically allow anybody to describe and implement their evolutionary algorithm without having the difficult task to program the algorithm itself. By specifying only application-dependent functions and parameters, the user gets the source code for a full evolutionary algorithm, possibly implementing efficient methods such as CMA-ES with no effort. The `-cuda` option has been added to the compiler to parallelize the algorithm over a GPGPU card.

5.1 Standard EASEA Evolutionary Algorithm

The EASEA compiler uses basic functions that are implemented in the EASEA library. The EASEA language [1] uses a user readable template of an algorithm, that is used to generate source code that incorporates the user functions, and some evolutionary engine specifications (such as the kind of selectors to be used for parent selection, population size, ...).

The evolutionary engine is kept on the CPU, that has the task of generating a population of individuals to evaluate (initial population, and subsequently children population). The evaluation function is compiled for the GPU card using the `nvcc` (nVidia C Compiler) compiler and transferred on the GPU card. Then, as soon as a new population is ready, it is transferred onto the GPU where all individuals get evaluated in parallel (cf. fig. 1).

In a standard algorithm, an array containing the fitness values is sent back to the CPU and the evolutionary loop can continue. In the case of a memetic algorithm, evaluation of a child incorporates a local search that can modify the

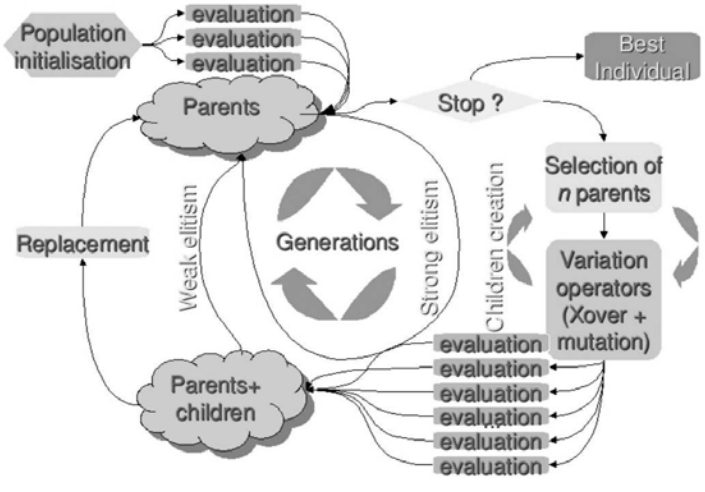


Fig. 1. Flowchart of the parallel evolutionary algorithm generated by EASEA

child's genome. This means that the whole population is sent back to the CPU along with the last evaluation of each individual.

5.2 Implementation

The implementation of a memetic algorithm on GPGPU was done in three steps, using the EASEA compiler:

1. The first step was to generate a standard algorithm with parallel evaluation over GPGPU using the `-cuda` option (that was developed for [6]). Then, the generated code was modified by hand to implement the memetic algorithm.
2. The second step was to optimize and minimize the changes done to the code.
3. Finally, the resulting modifications were integrated back into the EASEA library so that the EASEA language was able to automatically output the modified code, for a memetic algorithm with parallel evaluation on GPGPU, out of the same original `.ez` EA specification.

The major change that was applied to the standard generated algorithm was to implement a population transfer from the GPU memory back to the CPU memory, so that the evolutionary algorithm running on the CPU could continue with the modified children (in the "standard" parallel evolutionary algorithm, only an array containing the fitnesses was sent back to the CPU).

5.3 Local Search Algorithm

Since GPGPU cards are not fitted with a random number generator, a deterministic local search algorithm was used, although pseudo-random generators have been efficiently implemented on GPGPUs [4].

The local search algorithm used for the experiments requires a specific step and a specific number of search iterations. Until the maximum number of iterations is reached, the algorithm adds the step value to the first dimension of the individual, then evaluates the individual and compares its fitness to the fitness of the best individual to date. If the fitness improved, the individual replaces the best one and one step is added to the same dimension until the fitness stops improving, in which case the next dimension is explored in the same way. If, after the first attempt on one dimension, the fitness did not improve, the algorithm starts looking in the opposite direction.

Once the algorithm has browsed through all the dimensions, it goes back to the first dimension and repeats the process again until the specified number of iterations has been reached.

This algorithm is very crude, in that the step size is not adaptive, for instance. But the aim of this study is not to find the best local search algorithm that would fit all problems, but to experiment a memetic algorithm on a GPGPU card.

Other local search algorithms were tested during the development process but no impact on speedup has been detected. Therefore, all presented results use the simple algorithm described above.

6 Experiments

Experiments have been performed on "one half" of a GTX295 nVidia card *vs* a 3.6GHz Pentium IV with 3GB RAM under linux 2.6.27 32 bits with nVidia driver 190.18 and CUDA 2.3. By "one half," we mean that only one of the 2 GPUs that are present on a GTX295 card has been used. This choice was made in order to have a one to one comparison: one GPU chip vs one core of a CPU.

Timings have been performed with the `gettimeofday()` POSIX function. Speedups have been measured on the whole algorithm or on the evaluation function only. When only the evaluation function was timed, what was really done is that the evaluation function only was timed on the CPU, while on the GPU, population transfer time to and from the GPU was added for fairness.

6.1 Tests on the Rosenbrock Benchmark

In this paper, the purpose of the experiments was not to test the efficiency of the local search algorithm, but rather to measure the speedup that parallelizing the local search would bring.

The first set of experiments were performed on the Rosenbrock function because it is very fast to compute. The idea was to expose the incurred overheads as much as possible so as to obtain the worst possible results and get a fair idea on the advantages of parallelizing the optimisation on a GPGPU card. Using an evaluation function that was much longer to evaluate would have hidden away the inevitable overhead.

Rosenbrock's function [9] can be defined by the following equation :

$$f(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N/2} \left[100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2 \right]$$

where N represents the number of dimensions, and therefore the genome size.

6.2 Speedups on Evaluation Time Only

Fig 2 shows the obtained speedup for (evaluation time + population transfer time) to and from the GPU *vs* (evaluation time only) on the Intel CPU, *i.e.* all this without the evolutionary algorithm.

Maximum speedup reaches a plateau above $\times 120$ for a population size of 32K individuals and as few as 256 iterations of the local search function.

Maximum speedup is attained for 2,048 and more individuals because under this population size, the cores of the GPU card are not fully loaded.

A "reasonably" good speedup of $\times 58$ is obtained for 2,048 individuals and 256 iterations but it is important to remember that this very fast benchmark function maximises the influence of overhead. The surface seems to rise steeply still afterwards, but this impression is given by the fact that the scales are logarithmic. It requires 16K individuals to obtain a $\times 115$ speedup, *i.e.* approximately only a double speedup (over $\times 58$) for 8 times the population size.

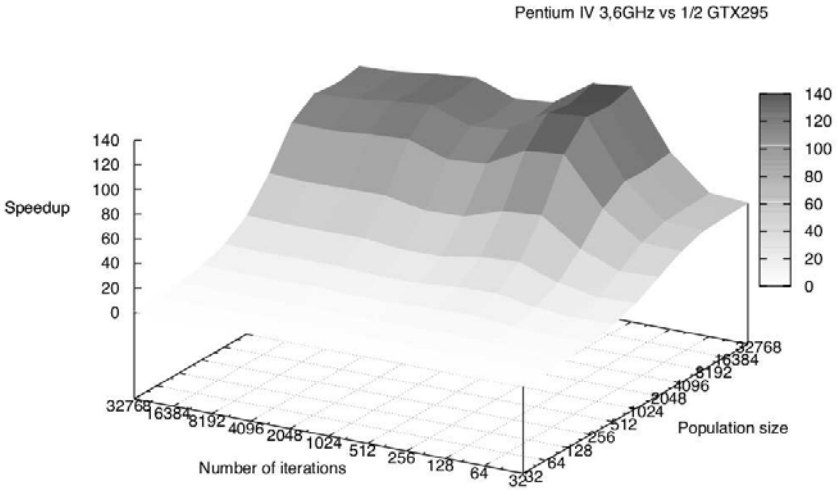


Fig. 2. Speedup for evaluation+transfer time only

Maximum speedup (×120) is obtained for 32K individuals and 256 iterations.

No reasonable explanation was found for the "pass" observed for 1,024 and 2,048 iterations above 8K individuals.

Since the power of GPU cards comes from their parallel architecture, one must use large populations to benefit from it.

6.3 Speedups on the Complete Memetic Evolutionary Algorithm

Fig 3 shows the obtained speedup for the complete memetic algorithm (and not evaluation time only) automatically created by the EASEA language.

Maximum speedup reaches a plateau at around ×91 for a population size of 32K individuals and 32K iterations of the local search function.

As above, speedup is not considerable until 2,048 individuals, because under this population size, the cores of the GPU card are not fully loaded.

A "reasonably" good speedup of ×47 is obtained for 2048 individuals and 2048 iterations. Much larger numbers are required in order to overcome the overhead of the evolutionary algorithm that runs on the CPU. Maximum speedup (×95) is obtained for 32K individuals and 16K iterations.

6.4 Influence of Population Transfer Time and Genome Size

Fig. 4 (top) shows evaluation time for 32K individuals and 32K iterations, w.r.t. the number of dimensions of the problem (×4 to get the genome size in bytes). Eval. time ranges from 3.44s for 16 dimensions to 238.9s for 1,024 dimensions.

Fig. 4 (bottom) shows the measured transfer time of the whole population towards the GPU and back for 32K individuals. GPU cards are designed to

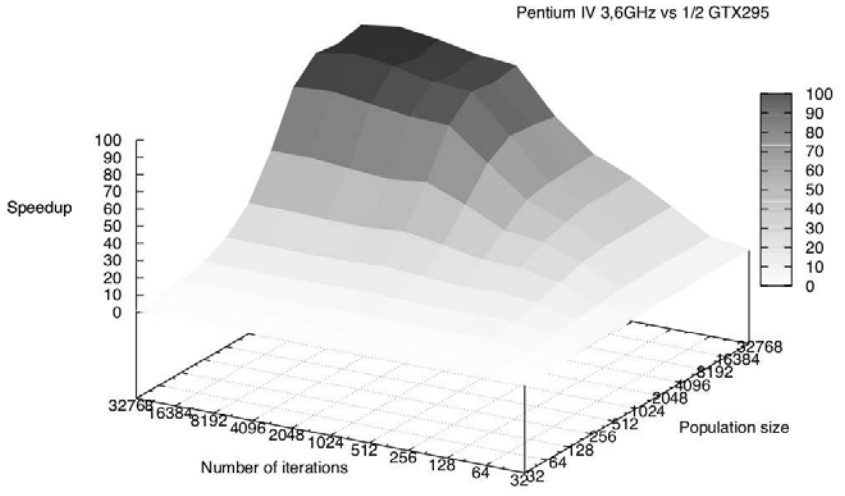


Fig. 3. Speedup for the complete algorithm

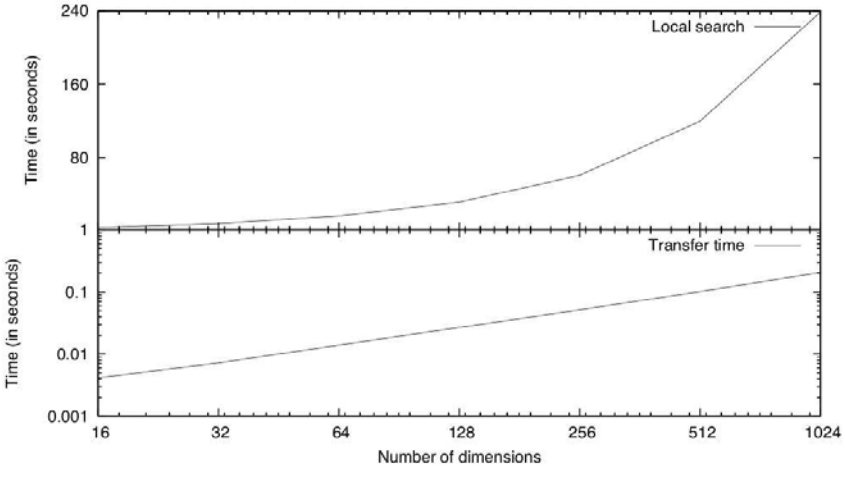


Fig. 4. Influence of transfer time and genome size

transfer millions of pixels per image frame, so throughput is huge (0.2 seconds for 32K individuals of 1,024 dimensions, *i.e.* 4KBytes).

Transfer time of the population to the GPU and back is negligible.

6.5 Real World Experiment

Evaluating speedup on a benchmark such as Rosenbrock is great, but this gets nowhere near the kind of evaluation function that is used in real world problems.

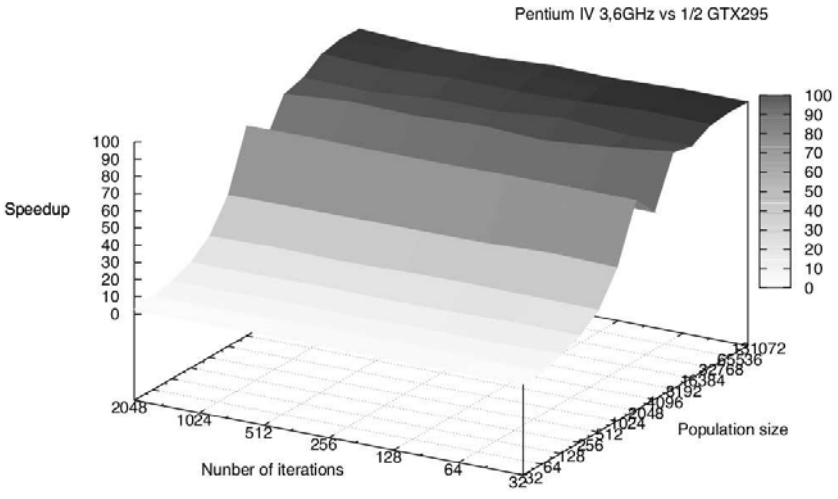


Fig. 5. Speedup on a real-world 12 dimensions problem

Therefore, the same setup was tested on a real chemistry example, where the 400 lines fitness function evaluates the probability for a crystal structure to be a zeolite. The genome is made of 12 floats (x, y, z position of 4 atoms) and the fitness function is in fact a composition of 4 sub-functions:

$$F(a_x, a_y, a_z, \ldots, d_x, d_y, d_z) = (F_a + F_b + F_c)/1000 * (1 + F_d)$$

Fa, Fb, Fc, and Fd are functions of atoms distances, angles, mean angles, and connectivity respectively [2].

The fitness function takes much more time to evaluate than the Rosenbrock function, meaning that the number of iterations does not much impact speedup, and overhead is minimized, which accounts for a more predictible surface: 32 iterations are sufficient to keep the GPU cores busy (where many more iterations were needed for the ultra-fast Rosenbrock function). The quality of the results is discussed in a chemistry paper.

Therefore, only population size has an influence on speedup. As usual, the card needs at least 2048 individuals to get a "reasonable" ×71 speedup for only 32 iterations. Then, adding more individuals allows the card to optimize its scheduling, and a maximum speedup of ×94 is obtained for 65K individuals and 64 iterations (to be compared with ×91 for 65K individuals and 32 iterations, or ×84 obtained with 16K individuals only and 32 iterations).

We have no explanation for the drop for 4096 individuals.

7 Conclusion and Future Work

This paper clearly shows how memetic algorithms (evolutionary algorithms coupled with a local search function) can fully benefit from being executed on highly

parallel GPGPU cards that were originally designed to execute the same rendering algorithm in parallel on millions of different pixels / vertices.

As soon as a population size of 2,048 individuals is reached, a speedup of $\approx \times 47$ is attained compared to a 3.6GHz Pentium 4 CPU, rising up to $\approx \times 120$ on a very simple benchmark function (Rosenbrock) or $\times 91$ on a real world problem, but the price to pay to maximise scheduling is to use a *huge* population.

Note that for the sake of simplicity these speedups were obtained on one GPU chip of a dual-GPU GTX295 nVidia card vs one core of a CPU. Tests with a memetic algorithm on the full card have not been performed yet, but preliminary tests on Genetic Programming show that using several cards is not much of a problem. However, using both GPUs of the GTX295 card will certainly imply doubling the population size for a double speedup. Other nVidia cards can be used, with different speedups.

PC motherboards exist that can host 4 such cards, so speedups of around $\times 960$ can be envisaged compared to a single core of an INTEL CPU, meaning that a one day computation on such a machine would be equivalent to several years's computation on a regular PC. This could allow to tackle problems that are up to now intractable.

Finally, (and this is a big part of the presented work), the presented GPU parallelization has been included in the EASEA language. The same `rosenbrock.ez` file will compile for CPU only if no option is given on the `easea` compiler command line. If the computer contains a GTX295 card, the presented results will be obtained if the `rosenbrock.ez` file is compiled with the `-cuda` option.

Future work consists in implementing other evolutionary paradigms onto GPGPU cards and to include them in the EASEA language. Doing this will allow the EC community to use these great cards without the need to understand how to program them. This is very important for evolutionary computation, that can use such massive parallelization.

References

1. 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)
2. Corma, A., Moliner, M., Serra, J.M., Serna, P., Diaz-Cabanas, M.J., Baumes, L.A.: A new mapping/exploration approach for ht synthesis of zeolites. Chemistry of Materials, 3287–3296 (2006)
3. Hart, W.E., Krasnogor, N., Smith, J.E.: Recent Advances in Memetic Algorithms. Springer, Heidelberg (2005)
4. Langdon, W.B.: A fast high quality pseudo random number generator for graphics processing units. In: Wang, J. (ed.) 2008 IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6, pp. 459–465. IEEE, Los Alamitos (2008)
5. Luo, Z., Liu, H.: Cellular genetic algorithms and local search for 3-SAT problem on graphic hardware. In: IEEE Congress on Evolutionary Computation CEC 2006, pp. 2988–2992 (2006)
6. Maitre, O., Baumes, L.A., Lachiche, N., Corma, A., Collet, P.: Coarse grain parallelization of evolutionary algorithms on gpgpu cards with easea. In: GECCO, pp. 1403–1410 (2009)

7. Munawar, A., Wahib, M., Munetomo, M., Akama, K.: Hybrid of genetic algorithm and local search to solve max-sat problem using nvidia cuda framework. Genetic Programming and Evolvable Machines 10(4), 391–415 (2009)
8. Maitre, O., Lachiche, N., Collet, P.: Fast evaluation of GP trees on GPGPU by optimizing hardware scheduling. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Şima Uyar, A. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 301–312. Springer, Heidelberg (2010)
9. Shang, Y.-W., Qiu, Y.-H.: A note on the extended rosenbrock function. Evol. Comput. 14(1), 119–126 (2006)
10. Wong, M., Wong, T.: Parallel hybrid genetic algorithms on Consumer-Level graphics hardware. In: IEEE Congress on Evolutionary Computation, CEC 2006, pp. 2973–2980 (2006)

Advancing Model–Building for Many–Objective Optimization Estimation of Distribution Algorithms

Luis Martí, Jesús García, Antonio Berlanga, and José M. Molina

Universidad Carlos III de Madrid, Group of Applied Artificial Intelligence
Av. de la Universidad Carlos III, 22. Colmenarejo, Madrid 28270, Spain
{lmarti,jgherrer}@inf.uc3m.es, {aberlan,molina}@ia.uc3m.es

Abstract. In order to achieve a substantial improvement of MOEDAs regarding MOEAs it is necessary to adapt their model–building algorithms. Most current model–building schemes used so far off–the–shelf machine learning methods. These methods are mostly error–based learning algorithms. However, the model–building problem has specific requirements that those methods do not meet and even avoid.

In this work we dissect this issue and propose a set of algorithms that can be used to bridge the gap of MOEDA application. A set of experiments are carried out in order to sustain our assertions.

1 Introduction

The multi–objective optimization problem (MOP) can be expressed as the problem in which a set of objective functions should be jointly optimized. In this class of problems the optimizer must find one or more feasible solutions that jointly minimizes (or maximizes) the objective functions. Therefore, the solution to this type of problem is a set of trade–off points.

Evolutionary algorithms (EAs) have proven themselves as a valid and competent approach from theoretical and practical points of view. These multi–objective evolutionary algorithms (MOEAs) [4] have succeeded when dealing with these problems because of their theoretical properties and real–world application performance.

There is a class of MOPs that are particularly appealing because of their inherent complexity: the so–called many–objective problems [18]. These are problems with a relatively large number of objectives.

The results of works that have ventured into these problems called for the search of other approaches that could handle many–objective problems with a reasonable performance. Among such approaches we can find estimation of distribution algorithms (EDAs) [13]. However, although multi–objective EDAs (MOEDAs) have yielded some encouraging results, their introduction has not lived up to their a priori expectations. This fact can be attributed to different causes, some of them, although already existing in single–objective EDAs, are better exposed in MOEDAs, while others are derived from the elements taken

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 512–521, 2010.

from MOEAs. An analysis on this issue led us to distinguish a number of inconveniences, in particular, the drawbacks derived from the incorrect treatment of population outliers; the loss of population diversity, and; the dedication of an excessive computational effort to finding an optimal population model.

There have been some works that have dealt with those three issues, in particular with the loss of diversity. Nevertheless, the community has failed to acknowledge that, perhaps, the underlying cause for those problems can be traced back to the algorithms used for model–building in EDAs.

In this work we examine the model–building issue of EDAs in order to show that some its characteristics, which have been ignored so far, render most current approaches inviable. We hypothesize that the problems of current EDAs can be traced back to the error–based machine learning algorithms used for mode–building and, that new classes of algorithms must be applied to properly deal with the problem. With that idea in mind we carried out a set of experiments that compare some algorithms typically used for model–building with other that, according to our hypothesis should perform well in this class of problems.

Reaching a rigorous understanding of the state–of–the–art in MOEDAs' model–building is hard since each model builder is embedded in a different MOEDA framework. Therefore, in order to comprehend the advantages and shortcomings of each algorithm, they should be tested under similar conditions and isolated from the MOEDA it is part of. That is why, in this work we assess some of the main machine learning algorithms currently used or suitable for model–building in a controlled environment and under identical conditions. This framework guarantees the direct comparison of the algorithms and allows for valid tests.

The rest of this contribution proceeds as we introduce the theoretical aspects that support our discussions. We then deal with the model–building problem, its properties and how it has been approached by the main MOEDAs. Subsequently, a set of experiments, using community–accepted, complex and scalable test problems with a progressive increase in the number of objective functions. Finally some concluding remarks and lines for future work are put forward.

2 Theoretical Background

The concept of multi–objective optimization refers to the process of finding one or more feasible solutions of a problem that corresponds to the extreme values (either maximum or minimum) of two or more functions subject to a set of restrictions.

More formally, a multi–objective optimization problem (MOP) can be defined as:

Definition 1 (Multi–objective Optimization Problem)

$$\left.\begin{array}{c} \text{minimize } \boldsymbol{F}(\boldsymbol{x}) = \langle f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x}) \rangle \,, \\ \text{with } \boldsymbol{x} \in \mathcal{D} \,, \end{array}\right\} \tag{1}$$

where $\mathcal{D}$ is known as the decision space. *The functions $f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})$ are the* objective functions. *The image set, $\mathcal{O}$, product of the projection of $\mathcal{D}$ through $f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})$ is called* objective space *($\boldsymbol{F} : \mathcal{D} \rightarrow \mathcal{O}$).*

In this class of problems the optimizer must find one or more feasible solutions that jointly minimizes (or maximizes) the objective functions. Therefore, the solution to this type of problem is a set of trade–off points. The adequacy of a solution can be expressed in terms of the Pareto dominance relation. The solution of (1) is the Pareto–optimal set, $\mathcal{D}^*$; which is the subset of $\mathcal{D}$ that contains elements that are not dominated by other elements of $\mathcal{D}$. Its image in objective space is called Pareto–optimal front, $\mathcal{O}^*$.

MOPs have been addressed with a broad range of approaches. Among them, evolutionary algorithms (EAs) have proven themselves as a valid and competent approach from theoretical and practical points of view. These multi–objective evolutionary algorithms (MOEAs) have succeeded when dealing with these problems because of their theoretical properties and real–world application performance.

2.1 Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDAs) have been claimed as a paradigm shift in the field of evolutionary computation. Like EAs, EDAs are population based optimization algorithms. However in EDAs the step where the evolutionary operators are applied to the population is substituted by construction of a statistical model of the most promising subset of the population. This model is then sampled to produce new individuals that are merged with the original population following a given substitution policy. Because of this model–building feature EDAs have also been called probabilistic model–building genetic algorithms (PMBGAs).

The introduction of machine learning techniques implies that these new algorithms lose the biological plausibility of its predecessors. In spite of this, they gain the capacity of scalably solve many challenging problems, significantly outperforming standard EAs and other optimization techniques.

Probably because of their success in single–objective optimization, EDAs have been extended to the multi–objective optimization problem domain, leading to multi–objective EDAs (MOEDAs) [17].

3 Error–Based Learning in Model–Building Algorithms

One topic that remains not properly studied inside the MOEDA scope is the scalability of the algorithms. The most critical issue is the dimension of the objective space. It has been experimentally shown to have an exponential relation with the optimal size of the population. This fact implies that, with the increase of the number of objective functions an optimization algorithm needs an exponential amount of resources made available to it.

Notwithstanding the diverse efforts dedicated to providing usable model–building methods for EDAs the nature of the problem itself has received relatively low attention. In spite of the progressively improving succession of results of EDAs, one question arises when looking for ways to further improve them. Would current statistically sound and robust approaches be valid for the problem being addressed? or, in other terms, does the model–building problem have particular demands that require custom–made algorithms to meet them? Machine learning and statistical algorithms, although suitable for their original purpose, might not be that effective in the particular case of model–building.

Generally, those algorithms are off–the-shelf machine learning methods that were originally intended for other classes of problems. On the other hand, the model–building problem has particular requirements that those methods do not meet and even have conflicts with. Furthermore, the consequences of this misunderstanding would be more dramatic when scaling up in the amount of objectives since the situation is aggravated by the implications of the curse of dimensionality.

An analysis of the results yielded by current multi–objective EDAs and their scalability with regard to the number of objective leads to the identification of certain issues that might be hampering the obtention of substantially better results with regard to other evolutionary approaches. Among those issues we can distinguish the following: incorrect treatment of data outliers, and; loss of population diversity.

This behavior, in our opinion, can be attributed the error–based learning approaches that take place in the underachieving MOEDAs. Error–based learning is rather common in most machine learning algorithms. It implies that model topology and parameters are tuned in order to minimize a global error measured across the learning data set. This type of learning isolated data is not taken into account because of their little contribution to the overall error and therefore they do not take an active part of learning process. In the context of many problems this behavior makes sense, as isolated data can be interpreted as spurious, noisy or invalid data.

That is not the case of model–building. In model–building all data is equally important and, furthermore, isolated data might have a bigger significance as they represent unexplored zones of the current optimal search space. This assessment is supported by the fact that most the approaches that had a better performance do not follow the error–based scheme. That is why, perhaps another class of learning, like instance–based learning [11] or match–based learning [7] would yield a sizable advantage. Therefore, it can be presumed that, in order to obtain a substantial improvement on this matter, algorithms that conform those types of learning should be applied.

3.1 Randomized Leader Algorithm

The randomized leader algorithm [8] is a fast and simple partitioning instance–based algorithm that was first used in the EDA context as part of the IDEA framework [3]. Its use is particularly indicated in situations when the overhead introduced by the clustering algorithm must remain as low as possible. Besides

its small computational footprint, this algorithm has the additional advantage of not having to explicitly specify in advance how many partitions should be discovered. On the other hand, the drawbacks of the leader algorithm are that it is very sensitive to the ordering of the samples and that the values of its thresholds must be guessed a priori and are problem dependent.

The algorithm goes over the data set exactly once. For each sample drawn it finds the cluster whose leader is the closest, given threshold the ρ_{Ld}. If such partition can not be found, a new partition is created containing only this single sample. Once the amount of samples in a cluster have exceeded the amount ρ_{Lc}, the leader is substituted by the mean of the cluster members. The mean of a partition changes whenever a sample is added to that partition. After obtaining the clustering a Gaussian mixture is constructed relying on it, as described for the naïve MIDEA algorithm [3]. This allows the sampling of the model in order to produce new elements.

3.2 Model–Building Growing Neural Gas

The model–building growing neural gas network (MB–GNG) [15] has been proposed as a form of dealing with the model–building issue. It has been devised with to deal with the model–building issue. The multi–objective neural EDA (MONEDA) [14], that incorporates MB–GNG, has yielded relevant results [14,16].

MB–GNG is a modified growing neural gas (GNG) network [6]. GNG networks have been chosen previously presented as good candidates for dealing with the model–building issue because of their known sensibility to outliers [19].

The network grows to adapt itself automatically to the complexity of the dataset being modelled. It has a fast convergence to low distortion errors and incorporates a cluster repulsion term to the original adaptation rule that promotes search and diversity.

3.3 Gaussian Adaptive Resonance Theory Network

Adaptive Resonance Theory (ART) neural networks are capable of fast, stable, on-line, unsupervised or supervised, incremental learning, classification, and prediction following a match–based learning scheme [7]. During training, ART networks adjust previously–learned categories in response to familiar inputs, and creates new categories dynamically in response to inputs different enough from those previously seen. A vigilance test allows to regulate the maximum tolerable difference between any two input patterns in a same category. It has been pointed out that ART networks are not suitable for some classes of classical machine–learning applications [20], however, what is an inconvenience in that area is a feature in our case.

There are many variations of ART networks. Among them, the Gaussian ART [21] is most suitable for model–building since it capable of handling continuous data. The result of applying Gaussian ART is a set of nodes each representing a local Gaussian density. These nodes can be combined as a Gaussian mixture that can be used to synthesize new individuals.

4 Experimental Analysis

To identify the model–building issue and its relation to error–based learning it is helpful to devise a comparative experiment that casts light on the different performance of a selected set of model–building algorithms subject to the same conditions when dealing with a group of problems of scaling complexity. In particular, we deal with two of the problems of the Walking Fish Group (WFG) continuous and scalable problem set [9], in particular the WFG4 and WFG9.

WFG4 is a separable and strongly multi–modal problem while WFG9 is non–separable, multi–modal and have deceptive local–optima. Both problems have a concave Pareto–optimal front that lies in the first orthant of a unit hypersphere located at the coordinates origin. This feature make them suitable for high–dimensional experiments where assessing the progress of algorithms is expensive for other shapes of fronts.

A MOEDA framework is shared by the model–building algorithms involved in the tests in order to ensure the comparison and reproducibility of the results.

The model–building algorithms involved in the tests were: (i) expectation maximization algorithm, as described for MIDEA [3]; (ii) Bayesian networks, as used in MrBOA [1]; (iii) $(1 + \lambda)$–CMA–ES as described in [10]; (iv) randomized leader algorithm, (v) MB–GNG, and; (vi) Gaussian ART.

4.1 Shared EDA Framework

To test MB–GNG is it essential to insert it in an EDA framework. This framework should be simple enough to be easily understandable but should also have a sufficient problem solving capacity. It should be scalable and preserve the diversity of the population.

Our EDA employs the fitness assignment used by the NSGA–II algorithm [5] and constructs the population model by applying MB–GNG. The NSGA–II fitness assignment was chosen because of its proven effectiveness and its relative low computational cost.

It maintains a population of individuals, P_t, where t is the current iteration. It starts from a random initial population P_0 of z individuals. It then proceeds to sort the individuals using the NSGA–II fitness assignment function. A set $\hat{P}_t$ containing the best $\lfloor \alpha \, |P_t| \rfloor$ elements is extracted from the sorted version of P_t,

$$\left| \hat{P}_t \right| = \alpha \, |P_t| \, . \tag{2}$$

The population model is then built using $\hat{P}_t$. The model is then used to create $\lfloor \omega \, |P_t| \rfloor$ new individuals is synthesized. Each one of these individuals substitute a randomly selected ones from the section of the population not used for model–building $P_t \setminus \hat{P}_t$. The set obtained is then united with best elements, $\hat{P}_t$, to form the population of the next iteration P_t.

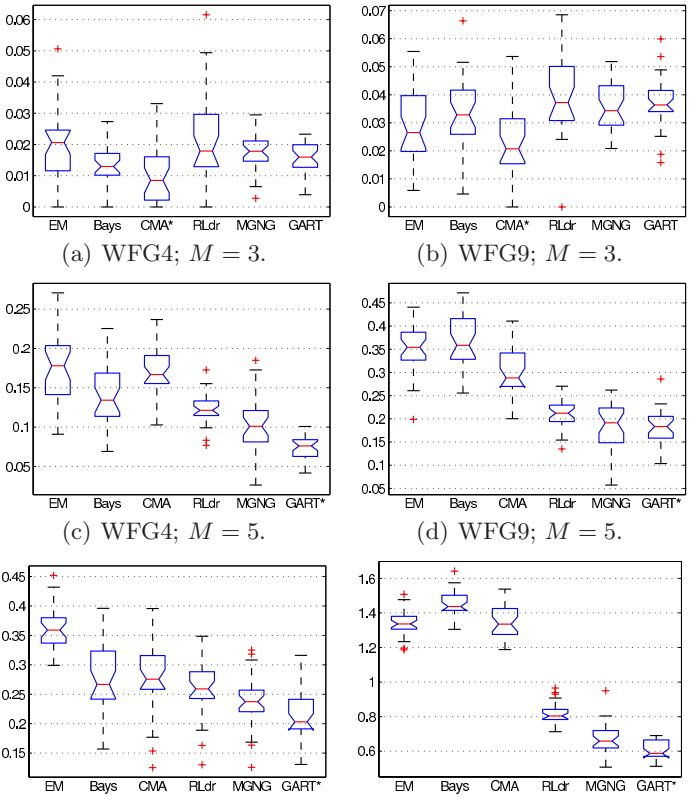
Iterations are repeated until a given stopping criterion is met. The output of the algorithm is the set of non–dominated individuals of P_t.

4.2 Results

The WFG4 and WFG6 problems were configured with 3, 5 and 7 objective functions. The dimension of the decision space was set to 10. Tests were carried out under the PISA experimental framework [2]. The binary additive epsilon indicator [12] was used to assess the performance. Although many other suitable indicators exist we have limited to this one because its low computational footprint and the space constraints imposed to this paper.

Figure 1 shows the box plots obtained after 30 runs of each algorithm for solving the different the problem/dimension configuration.

In the three dimensional problems our approach performed similarly to the rest of the algorithms. This was an expected outcome. However, in the case of

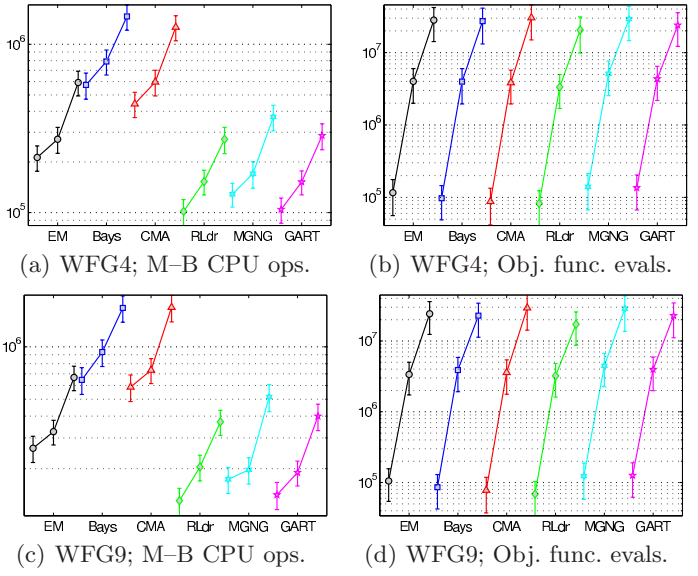


(a) WFG4; $M = 3$. (b) WFG9; $M = 3$.

(c) WFG4; $M = 5$. (d) WFG9; $M = 5$.

(e) WFG4; $M = 7$. (f) WFG9; $M = 7$.

Fig. 1. Boxplots of the binary additive epsilon indicator values obtained when dealing with the WFG4 and WFG9 problems with EDAs using expectation–maximization (EM), Bayesian networks (Bays), covariance matrix adaptation ES (CMA), randomized leader algorithm (RLdr), modified growing neural gas networks (MGNG) and Gaussian adaptive resonance theory neural networks (GART) for model–building. The result of each algorithm is measured against a sampled version of the Pareto–optimal front.



(a) WFG4; M–B CPU ops. (b) WFG4; Obj. func. evals.

(c) WFG9; M–B CPU ops. (d) WFG9; Obj. func. evals.

Fig. 2. Analysis of the computational cost of dealing with WFG4 and WFG9. The number of CPU ops dedicated for model–building and the number of objective functions evaluations are measured in each case (see Fig. 1 for algorithms' acronyms).

five and seven the three non–error–based learning algorithms outperform the rest of the optimizers applied.

One can hypothesize that, in this problem, the model–building algorithm induces the exploration of the search space and therefore it manages to discover as much as possible of the Pareto–optimal front. It is most interesting that our proposal exhibits rather small standard deviations. This means that it performed consistently well across the different runs. These results must be investigated further to understand if the low dispersion of the error indicators can only be obtained in the problems solved or if can be extrapolated to other problems.

These results are further confirmed by inspecting figure 2. Here the advantages of using error–based learning in approximation quality terms are supplemented by the low computational costs of those algorithms. It can be perceived that, while all algorithms used similar numbers of objective function evaluations, the three non–error-based ones required far less computational resources to build the population models.

It is important to underline the performance of Gaussian ART that had never been used before in this application context. Gaussian ART outperformed the rest of the approaches in 5 and 7 objectives in WFG4 and WFG6 in terms of solution quality and computational cost.

5 Conclusions

In this paper we have discussed an important issue in current evolutionary multi–objective optimization: how to build algorithms that have better scalability with regard to the number of objectives. In particular, we have focused on one promising set of approaches, the estimation of distribution algorithms.

We have argued that most of the current approaches do not take into account the particularities of the model–building problem they are addressing and, because of that they fail to yield results of substantial quality.

In any case, it seems obvious after the previous discussions and experiments that the model–building problem deserves a different approach that takes into account the particularities of the problem. Perhaps the ultimate solution to this issue is to create custom–made algorithms that meet the specific requirement of this problem.

Acknowledgements

This work was supported by projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB, CAM CONTEXTS S2009/TIC-1485 and DPS2008-07029-C02-0.

References

1. Ahn, C.W.: Advances in Evolutionary Algorithms. Theory, Design and Practice. Springer, Heidelberg (2006)
2. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
3. Bosman, P.A.N., Thierens, D.: The naïve MIDEA: A baseline multi–objective EA. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 428–442. Springer, Heidelberg (2005)
4. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. In: Genetic and Evolutionary Computation, 2nd edn. Springer, New York (2007), http://www.springer.com/west/home/computer/foundations?SGWID=4-156-22-173660344-0
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
6. Fritzke, B.: A growing neural gas network learns topologies. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Advances in Neural Information Processing Systems, vol. 7, pp. 625–632. 625–632. MIT Press, Cambridge (1995)
7. Grossberg, S.: Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control. Reidel, Boston (1982)
8. Hartigan, J.A.: Clustering Algorithms. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York (1975)

9. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
10. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation 15(1), 1–28 (2007)
11. Kibler, D., Aha, D.W., Albert, M.K.: Instance–based prediction of real–valued attributes. Computational Intelligence 5(2), 51–57 (1989)
12. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
13. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A new tool for Evolutionary Computation. In: Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)
14. Martí, L., García, J., Berlanga, A., Molina, J.M.: Introducing MONEDA: Scalable multiobjective optimization with a neural estimation of distribution algorithm. In: Thierens, D., Deb, K., Pelikan, M., Beyer, H.G., Doerr, B., Poli, R., Bittari, M. (eds.) GECCO 2008: 10th Annual Conference on Genetic and Evolutionary Computation, pp. 689–696. ACM Press, New York (2008); EMO Track "Best Paper" Nominee
15. Martí, L., García, J., Berlanga, A., Molina, J.M.: Scalable continuous multiobjective optimization with a neural network–based estimation of distribution algorithm. 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. 535–544. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-78761-7_59
16. Martí, L., García, J., Berlanga, A., Molina, J.M.: Solving complex high–dimensional problems with the multi–objective neural estimation of distribution algorithm. In: Thierens, D., Deb, K., Pelikan, M., Beyer, H.G., Doerr, B., Poli, R., Bittari, M. (eds.) GECCO 2009: 11th Annual Conference on Genetic and Evolutionary Computation. ACM Press, New York (2009) (to appear)
17. Pelikan, M., Sastry, K., Goldberg, D.E.: Multiobjective estimation of distribution algorithms. In: Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.) Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Studies in Computational Intelligence, pp. 223–248. Springer, Heidelberg (2006)
18. Purshouse, R.C., Fleming, P.J.: On the evolutionary optimization of many conflicting objectives. IEEE Transactions on Evolutionary Computation 11(6), 770–784 (2007), http://dx.doi.org/10.1109/TEVC.2007.910138
19. Qin, A.K., Suganthan, P.N.: Robust growing neural gas algorithm with application in cluster analysis. Neural Networks 17(8–9), 1135–1148 (2004), http://dx.doi.org/10.1016/j.neunet.2004.06.013
20. Sarle, W.S.: Why statisticians should not FART. Tech. rep., SAS Institute, Cary, NC (1995)
21. Williamson, J.R.: Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps. Neural Networks 9, 881–897 (1996)

Estimation Distribution Differential Evolution*

Ernesto Mininno and Ferrante Neri

Department of Mathematical Information Technology P.O. Box 35 (Agora) 40014
University of Jyväskylä, Finland
{ernesto.mininno,ferrante.neri}@jyu.fi

Abstract. This paper proposes a novel adaptation scheme for Differential Evolution (DE) frameworks. The proposed algorithm, namely Estimation Distribution Differential Evolution (EDDE), is based on a DE structure and employs randomized scale factor ad crossover rate values. These values are sampled from truncated Gaussian probability distribution functions. These probability functions adaptively vary during the optimization process. At the beginning of the optimization the truncated Gaussian functions are characterized by a large standard deviation values and thus are similar to uniform distributions. During the later stages of the evolution, the probability functions progressively adapt to the most promising values attempting to detect the optimal working conditions of the algorithm. The performance offered by the proposed algorithm has been compared with those given by three modern DE based algorithms which represent the state-of-the-art in DE. Numerical results show that the proposed EDDE, despite its simplicity, is competitive with the other algorithms and in many cases displays a very good performance in terms of both final solution detected and convergence speed.

1 Introduction

Differential Evolution (DE, see [1]) is an efficient function optimizer which has a good performance for diverse continuous optimization problems. Reasons for success of the DE can be found in its simplicity and ease of implementation, while at the same time demonstrating reliability and high performance. In addition, the fact that only three parameters require tuning greatly contributes to the rapid diffusion of DE schemes among computer scientists and practitioners. Although the DE undoubtedly has a great potential, setting of the control parameters is not a trivial task, since it has a heavy impact on the algorithmic performance. Thus, over the years, the DE community has intensively investigated the topic of parameter setting. Several studies have been reported, e.g. in [2], [3], and [4], and led to contradictory conclusions.

From an algorithmic viewpoint, reasons for the success of DE have been highlighted in [5]: success of DE is due to an implicit self-adaptation contained within

* This research is supported by the Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing and by Tekes - the Finnish Funding Agency for Technology and Innovation, grant 40214/08 (Dynergia).

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 522–531, 2010.

the algorithmic structure. More specifically, since, for each candidate solution, the search rule depends on other solutions belonging to the population, the capability of detecting new promising offspring solutions depends on the current distribution of the solutions within the decision space. During early stages of the optimization process, solutions tend to be spread out within the decision space. For a given scale factor value, this implies that the mutation appears to generate new solutions by exploring the space by means of a large step size when candidate solutions are, on average, distant from each other. During the optimization process, the solutions of the population tend to concentrate on specific parts of the decision space. Therefore, step size in the mutation is progressively reduced and the search is performed in the neighborhood of the solutions. In other words, due to its structure, a DE scheme is highly explorative at the beginning of the evolution and subsequently becomes more exploitative during optimization.

Although this mechanism seems at first glance to be very efficient, it hides a limitation. If for some reason, the algorithm does not succeed in generating offspring solutions which outperform the corresponding parent, the search is repeated again with similar step size values and will likely fail by falling into an undesired stagnation condition (see [6]). Stagnation is the undesired effect which occurs when a population-based algorithm does not converge to a solution (even suboptimal) and the population diversity is still high. In the case of the DE, stagnation occurs when the algorithm does not manage to improve upon any solution of its population for a prolonged number of generations. In other words, the main drawback of the DE is that the scheme has, for each stage of the optimization process, a limited amount of exploratory moves. If these moves are not enough for generating new promising solutions the search can be heavily compromised.

Thus, in order to enhance the DE performance, alternative search moves should support the original scheme and promote a successful continuation of the optimization process. For example, paper [7] propose a dynamic scale factor and a randomized scale factor (similar to jitter and dither [1]). Paper [8] proposes a controlled randomization of scale factor and crossover rate. Papers [9] and [10] propose the hybridization of a DE framework with a local search component. Paper [11] proposes a novel index-based neighborhood concept which enlarges the set of possible offspring generated. Paper [12] employs multiple mutation strategies coordinated by a memory based adaptive system. Paper [13] proposes a complex adaptive system for a randomized selection of control parameters.

In this fashion, the present paper proposes a novel, relatively simple, and efficient adaptive scheme for performing control parameter setting in DE frameworks. The proposed control scheme is inspired by Estimation Distribution Algorithms (EDAs), see [14]. Control parameters are sampled from truncated Gaussian Probability Distribution Functions (PDFs) which adapt during the optimization process towards the most promising values and attempt to follow the needs of the evolution. The resulting algorithm is here indicated as Estimation Distribution Differential Evolution (EDDE).

The remainder of this paper is organized in the following way. Section 2 describes the proposed algorithm. Section 3 gives the experimental setup and compares the performance of the proposed EDDE with three DE based algorithms, recently proposed in literature. Section 4 gives the conclusions of this work.

2 EDDE Algorithm

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

The proposed EDDE consists of the following steps. At the beginning of the optimization process, N_p individuals are randomly sampled. Each individual x_i is characterized by the following structure:

$$x_i = \langle x_{i,1}, x_{i,2}, ..., x_{i,j}, ...x_{i,n}, F_i, CR_i \rangle . \tag{1}$$

where $x_{i,j}$ $\forall j$ are the design variable sampled from the decision space D by means of a uniform distribution function; F_i and CR_i are control parameters (namely scale factor and crossover rate respectively) sampled from two truncated Gaussian PDFs. The first PDF is for sampling the scale factor F_i while the second is for sampling the crossover rate CR_i. The PDF related to the scale factors is initialized with mean value $\mu_F = 0.6$, truncation interval $[0, 1.2]$, and standard deviation $\sigma_F = 10$. The PDF related to the crossover rates is initialized with mean value $\mu_{CR} = 0.5$, truncation interval $[0, 1]$, and standard deviation $\sigma_{CR} = 10$. The initialization of σ values equal to 10 is done in order to simulate (at the initial stage) uniform distributions, see Figs. 1.

At each generation, for each individual x_i of the N_p, three individuals x_r, x_s and x_t are pseudo-randomly extracted from the population. A new scale factor F_{off} is sampled from its PDF in order to perform the mutation. According to the most classical DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{off} = x_t + F_{off}(x_r - x_s). \tag{2}$$

When the provisional offspring has been generated by mutation, each gene of the individual x'_{off} is exchanged with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{i,j} & \text{if } rand(0,1) < CR_{off} \\ x'_{off,j} & otherwise \end{cases} \tag{3}$$

where $rand(0, 1)$ is a uniformly sampled random number between 0 and 1, j is the index of the gene under examination, and CR_i is the crossover rate sampled from its PDF. The offspring is then composed of its design variables $x_{i,j}$ and the newly generated F_{off} and CR_{off}. The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning strategy, it replaces x_i at the end of the generation if and only if $f(x_{off}) \leq f(x_i)$; otherwise no replacement occurs.

If $f(x_{off}) \leq f(x_i)$ the PDFs related to F and CR are updated in order to increase the likelihood of selecting the control parameter values which allowed the generation of promising offspring. This operation is obtained by moving the mean values of PDFs towards the most promising parameter values and varying the standard deviation around the mean values. The standard deviation values are decreased when successful control parameters are generated close to the mean value and increased when a successful value is distant from the mean. This mechanism means that the adaptive system tends to focus on promising values but if the guess turns out to be incorrect the adaptive system attempts to explore again a larger set of values. Let us indicate with *winner* the scale factor (or crossover rate) related to the solution, between parent x_i and offspring x_{off}, characterized by the lowest fitness value and with *loser* the scale factor (crossover rate) related to the other solution. For each pairwise comparison, the values of μ_F, μ_{CR}, σ_F, and σ_{CR} are updated according to the following rules:

$$\mu^{k+1} = \mu^k + \frac{1}{N_p}(winner - loser), \tag{4}$$

and

$$\left(\sigma^{k+1}\right)^2 = \left(\sigma^k\right)^2 + \left(\mu^k\right)^2 - \left(\mu^{k+1}\right)^2 + \frac{1}{N_p}\left(winner^2 - loser^2\right). \tag{5}$$

where k is the comparison index. Since the update rules for F and CR have the same structure, formulas are reported (without pedex F or CR) only once. In addition, it must be remarked that although each pairwise comparison contributes to the update of the new truncated Gaussian PDF, the actual updates are performed all at once, at end of each generation, i.e. after N_p comparisons. Formulas (4) and (5) are derived from the study reported in [15] where the truncated Gaussian PDFs model the population of a compact Genetic Algorithm.

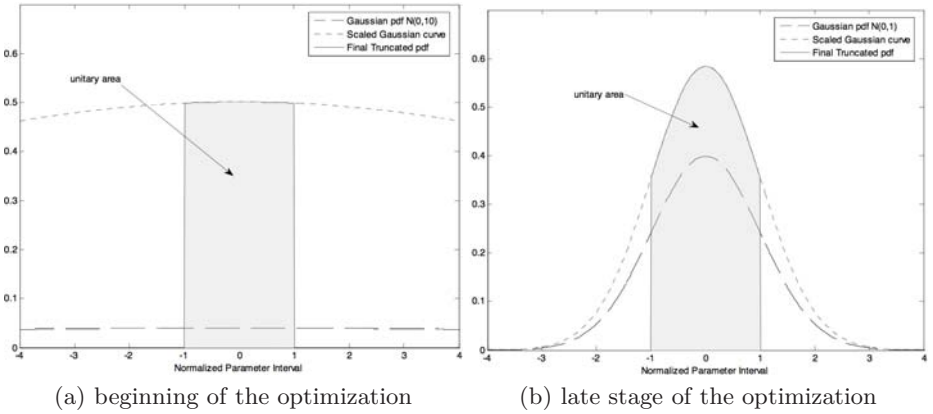
The main ideas behind the proposed EDDE are the following. As mentioned above the success of DE is due to its implicit self-adaptation. However, this mechanism often fails to converge to the optimum and typically leads to a situation characterized by the incapability of outperforming the best individual of the population despite a high diversity condition, see [6]. This fact can be justified from complementary perspectives. From the first perspective, DE has a too deterministic structure where the available search moves are generated by the possible combinations among population individuals. An increase in the randomization of a DE structure or the integration of randomized search within DE evolutionary loop has been widely applied in literature, see [7] and [8]. Looking at the same phenomenon from a complementary perspective, we can observe that DE has a limited amount of search moves. Thus, in order to enhance upon the performance of a DE algorithm extra moves are needed. This conclusion is supported by recent studies, e.g. [9], [10], and [16], where extra search moves with respect to those belonging to the DE logic are integrated within DE structure in order to outperform standard DE. Finally, some recently proposed algorithms combine randomization and extra search moves, see [12] and [13]. In addition,

although a proper parameter setting can help to have a good algorithmic performance, the fact that such setting is very problem dependant, see e.g. [4], makes the choice of parameters a difficult task. In addition, due to the dynamic nature of DE (as well as all metaheuristics), probably there is not an optimal parameter setting for a given problem since it varies during the optimization process.

In this light, the proposed EDDE takes into account the dynamic nature of DE and thus promotes a co-evolution of the algorithmic parameters. It can be noticed that the PDFs evolve with the population of solutions during optimization process. Thus, the F and CR values which seem to generate solutions with the most promising performance are more likely to be selected for the subsequent generations. When the parameter setting is not promising anymore during the evolution the adaptive system searches for new promising values by enlarging standard deviations. Since it cannot be established a priori the most promising values for scale factor and crossover rate and their optimal values does not seem to follow a clear trend during the evolution, the adaptive system is built up in order to follow the natural development of the optimal control parameter values. Moreover, F and CR are sampled from PDFs. Thus, a randomization is introduced into the search logic. This randomization is supposed to implicitly increase the search moves of the algorithm and to not fix control parameters values but allow a certain range of variability for their selection.

Figs. 1 give a graphical representation of the behavior of the truncated PDFs, at the beginning of the optimization and at a later stage of the process. The long-dashed line indicates the theoretical Gaussian PDF. The short-dashed line indicates the corresponding scaled Gaussian curve, i.e. the Gaussian PDF is scaled in order to obtain that its integral between two prefixed extremals (in our case -1 and 1 as shown in Figs. 1) is equal to one (unitary area). The solid line represents the truncated the final truncated PDF, i.e. that portion of the scaled gaussian curve between the two extremals.

Finally it must be highlighted that the EDDE version here presented is based on the so called rand/1/bin strategy described in formulas (2) and (3). Thus, the



(a) beginning of the optimization (b) late stage of the optimization

Fig. 1. Truncated gaussian PDFs

```
generate N_p individuals of the initial population and initial PDF pseudo-randomly
while budget condition do
  for i = 1 : N_p do
    compute f(x_i)
  end for
  k = 1
  for i = 1 : N_p do
    {** Mutation **}
    select three individuals x_r, x_s, and x_t
    sample F_off from PDF
    compute x'_off = x_t + F_off(x_r − x_s)
    {** Crossover **}
    x_off = x'_off
    sample CR_off from PDF
    for j = 1 : n do
      generate rand(0, 1)
      if rand(0, 1) < CR_off then
        x_off,j = x_i,j
      end if
    end for
    {** Selection **}
    if f(x_off) ≤ f(x_i) then
      save index for replacement x_i = x_off
    end if
    {** Update PDFs**}
    define winner and loser for F and CR
    for F and CR do
      μ^{k+1} = μ^k + (1/N_p)(winner − loser)
      (σ^{k+1})^2 = (σ^k)^2 + (μ^k)^2 − (μ^{k+1})^2 + (1/N_p)(winner^2 − loser^2)
    end for
    k = k + 1
  end for
  perform replacements
  perform updates of PDFs
end while
```

Fig. 2. Pseudo-code of EDDE/rand/1/bin

algorithm presented in this section can be indicated as EDDE/rand/1/bin. However, the proposed adaptation can be easily integrated within different mutation and crossover schemes, see e.g. [12].

For the sake of clarity, the pseudo-code highlighting working principles of the EDDE/rand/1/bin is shown in Fig. 2.

3 Experimental Results

The following test problems have been considered in this paper in order to verify the viability of the proposed approach.

1. Ackley's function:
$f_1(x) = -20 + e + 20\exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n} x_i^2}\right) + -\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)x_i\right)$,
with $n = 30$. Decision space $D = [-32, 32]^n$.
2. Michalewicz's function:
$f_2(x) = -\sum_{i=0}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2n}$,

with $n = 30$. Decision space $D = [0, \pi]^n$.
3. Pathological function:
$f_3(x) = \sum_{i=1}^{n-1} 1/2 + \left(\sin\left(\sqrt{100x_i^2 + x_{i+1}^2} - 1/2\right)\right) /$
$/ \left(1 + 0.001\left(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2\right)^2\right)$
with $n = 30$. Decision space $D = [-100, 100]^n$.
4. Tirronen's function:
$f_4(x) = 3\exp\left(-\frac{\|x\|^2}{10n}\right) - 10\exp\left(-8\|x\|^2\right) +$
$+\frac{2.5}{n}\sum_{i=1}^{n}\cos\left(5\left(x_i + (1 + i \bmod 2)cos\left(\|x\|^2\right)\right)\right)$
with $n = 30$. Decision space $D = [-5, 10]^n$.

The test problems f_5-f_8 have been obtained from f_1-f_4, by setting $n = 100$. In addition, four test problems here indicated with f_9, f_{10}, f_{11} and f_{12} have been generated from test problems f_2, f_4, f_6, f_8 respectively by applying a rotation operation. More specifically, these rotated problems have been generated through the multiplication of the vector of variables by a randomly generated orthogonal rotation matrix.

In order to perform a fair comparisons, three modern DE based algorithms have been considered in this study. More specifically, the proposed EDDE/rand/1 /bin, here indicated with EDDE for simplicity, has been compared with j-Differential Evolution (jDE) proposed in [8], J-Adaptive Differential Evolution (JADE) proposed in [13], and Self-Adaptive Differential Evolution (SADE) described in [12]. With reference to the notation contained in the original papers jDE has been run with $F_l = 0.1$, $F_u = 0.9$, and $\tau_1 = \tau_2 = 0.1$, JADE has been run with $c = 0.1$, SADE apart from the population size, which will be discussed later, is parameterless. In other words, each algorithm has been run with the parameter setting recommended by the original authors. EDDE has been run with the initial parameters on mean value and standard deviation indicated above. The population size N_p has been set equal to $2 \times n$ for all the algorithms under consideration. For each algorithm 50 independent runs have been performed. Each run has been stopped after $5000 \times n$ fitness evaluations.

Experimental results in terms of average final fitness and related standard deviation are given in Table 1. The best results are highlighted in bold face.

Table 1. Fitness ± standard deviation

Test Problem	jDE	JADE	SADE	EDDE
f_1	**5.773e-15 ± 1.76e-15**	2.007e+00 ± 9.14e-01	1.051e-14 ± 8.73e-15	5.921e-15 ± 1.79e-15
f_2	-2.917e+01 ± 2.39e-01	-2.147e+01 ± 5.15e-01	-2.641e+01 ± 1.52e+00	**-2.926e+01 ± 1.37e-01**
f_3	-9.273e+00 ± 1.72e+00	-5.752e+00 ± 1.47e+00	**-1.017e+01 ± 1.47e+00**	-1.004e+01 ± 1.33e+00
f_4	-2.499e+00 ± 8.42e-04	-2.364e+00 ± 2.05e-02	-1.504e+00 ± 4.37e-01	**-2.500e+00 ± 1.52e-07**
f_5	**2.354e-14 ± 4.30e-15**	7.481e+00 ± 2.84e+00	1.913e+00 ± 1.01e+00	1.253e+00 ± 3.61e-01
f_6	-8.876e+01 ± 1.37e+00	-3.973e+01 ± 1.07e+00	-7.625e+01 ± 4.76e+00	**-9.254e+01 ± 1.15e+00**
f_7	-2.357e+01 ± 3.12e+00	-1.164e+01 ± 4.27e+00	-1.682e+01 ± 5.77e+00	**-3.043e+01 ± 3.84e+00**
f_8	-2.473e+00 ± 8.05e-03	-2.143e+00 ± 2.29e-02	-1.056e+00 ± 6.61e-01	**-2.500e+00 ± 1.00e-15**
f_9	-9.338e+00 ± 1.79e+00	-9.406e+00 ± 1.86e+00	**-1.250e+01 ± 1.55e+00**	-1.094e+01 ± 1.30e+00
f_{10}	-2.246e+00 ± 1.95e-01	-2.023e+00 ± 1.72e-01	-2.373e+00 ± 1.12e-01	**-2.389e+00 ± 1.01e-01**
f_{11}	-1.201e+01 ± 2.93e+00	-1.323e+01 ± 2.84e+00	**-1.990e+01 ± 2.41e+00**	-1.456e+01 ± 2.63e+00
f_{12}	-2.149e+00 ± 1.92e-01	-2.186e+00 ± 2.96e-01	-2.381e+00 ± 8.37e-02	**-2.387e+00 ± 6.90e-02**

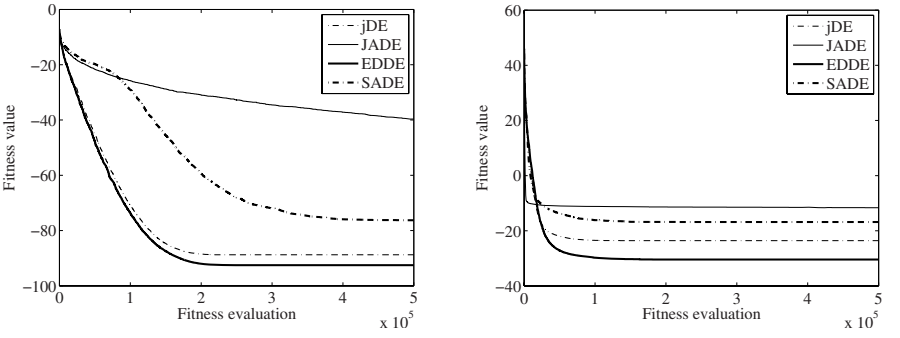
In order to carry out a numerical comparison of the convergence speed performance, for each test problem, the average final fitness value returned by the best performing algorithm G has been considered. Subsequently, the average fitness value at the beginning of the optimization process J has also been computed. The threshold value $THR = J - 0.95(J - G)$ has then been calculated. If an algorithm succeeds during a certain run to reach the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$ required, for each algorithm, to reach THR has been computed. Subsequently, the Q-test (Q stands for Quality) described in [5] has been applied. For each test problem and each algorithm, the Q measure is computed as $Q = \frac{\bar{ne}}{R}$ where the robustness R is the percentage of successful runs. For each test problem, the smallest value equals the best performance in terms of convergence speed. The value "∞" means that $R = 0$, i.e., the algorithm never reached the THR. Results of the Q-test are given in Table 2. In order to prove statistical significance of the results, the Wilcoxon Rank-sum test has been applied for a confidence level of 0.95. Table 2 shows results of the test. A "+" indicates the case in which EDDE statistically outperforms the other algorithm indicated in the heading of the column; a "=" indicates that the two algorithms have the same performance; a "-" indicates that EDDE is outperformed.

For the sake of clarity two examples of performance trend are shown in Fig. 3.

Numerical results show that EDDE is competitive with the other modern algorithms considered in this study. More specifically, results displayed in Table 1 show that EDDE obtains the best final value in eight cases out of the twelve considered. The statistical analysis carried out by means of Wilcoxon test confirms that in most cases EDDE significantly outperforms the other algorithms. As it can been observed in Table 2, over the thirty-six pairwise comparisons, EDDE is outperformed in only three cases and obtains similar results in seven cases. Thus, for the experimental setup considered in this paper, EDDE outperforms the other algorithms in 72.2% of the cases, has a comparable performance in 19.4% of the cases, and is outperformed in 8.3% of the cases. The performance in terms of convergence speed shows that EDDE obtains the best Q-measure values in eight cases out of the twelve considered. Thus, for this preliminary study we can conclude that EDDE is a very promising enhanced version of DE and

Table 2. Q-test and Wilcoxon Rank-Sum test

Test Problem	jDE	JADE	SADE	EDDE	jDE	JADE	SADE
f_1	624.6528	3116.6667	492.1007	**441.4062**	=	+	+
f_2	2528.4722	∞	22212.5	**1985.5903**	=	+	+
f_3	1566	16050	874.3945	**660.6481**	=	+	=
f_4	1742.5347	143200	46144.4444	**1293.1424**	+	+	+
f_5	1600.1736	∞	13061.1111	**12994.4444**	-	+	+
f_6	12501.2755	∞	∞	**6532.2049**	+	+	+
f_7	16027.7778	∞	44050	**2242.2902**	+	+	+
f_8	5534.2014	∞	74359.375	**3772.2222**	+	+	+
f_9	69075	29512.5	**4312.426**	38772.2222	+	+	-
f_{10}	12113	35312.5	**2111.4958**	5127.1468	+	+	=
f_{11}	∞	483600	**14649.2188**	∞	+	=	-
f_{12}	66631.9444	29487.6033	**5437.875**	13733.3678	+	+	=



(a) Performance trend on problem f_6 (b) Performance trend on problem f_7

Fig. 3. Examples of performance trend

seems able to compete with other modern DE based algorithms, representing the state-of-the-art in the field.

4 Conclusion

This paper proposes a novel adaptive scheme for Differential Evolution algorithms. This adaptation consists of a randomization of scale factor and crossover rate by means of truncated Gaussian Probability Distribution Functions and an update rule which modifies the parameters of these probability functions on the basis of the success of the most promising control parameters. The resulting algorithm seems robust and efficient over a set of various test problems. The proposed algorithm, despite its simplicity, offers a good performance compared to other modern sophisticated algorithms based on a Differential Evolution structure. Numerical results prove that the proposed algorithm is competitive for many test problems and outperforms the other algorithms considered in this study in most of considered cases. Future development of this work will consider possible memetic versions including local search components which follow an adaptive logic similar to the one proposed in this paper and an adaptive compact version of the algorithm here presented for implementation within micro-controllers and other systems presenting limited hardware features.

References

1. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
2. 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)
3. Price, K., Storn, R.: Differential evolution: A simple evolution strategy for fast optimization. Dr. Dobb's J. Software Tools 22(4), 18–24 (1997)

4. Zielinski, K., Weitkemper, P., Laur, R., Kammeyer, K.D.: Parameter study for differential evolution using a power allocation problem including interference cancellation. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1857–1864 (2006)
5. Feoktistov, V.: Differential Evolution. Springer, Heidelberg (2006)
6. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Ôsmera, P. (ed.) Proceedings of 6th International Mendel Conference on Soft Computing, pp. 76–83 (2000)
7. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 991–998. ACM, New York (2005)
8. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
9. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation 12(1), 107–125 (2008)
10. Neri, F., Tirronen, V.: Scale factor local search in differential evolution. Memetic Computing, 153–171 (2009)
11. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution with a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation 13(3), 526–553 (2009)
12. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13(2), 398–417 (2009)
13. Zhang, J., Sanderson, A.C.: Jade: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13(5), 945–958 (2009)
14. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Dordrecht (2001)
15. Mininno, E., Cupertino, F., Naso, D.: Real-valued compact genetic algorithms for embedded microcontroller optimization. IEEE Transactions on Evolutionary Computation 12(2), 203–219 (2008)
16. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation 12(1), 64–79 (2008)

Design of Continuous Controllers Using a Multiobjective Differential Evolution Algorithm with Spherical Pruning

Gilberto Reynoso-Meza, Javier Sanchis, Xavier Blasco, and Miguel Martínez

Instituto Universitario de Automática e Informática Industrial,
Universidad Politécnica de Valencia,
Camino de Vera s/n, 46022 Valencia, España
gilreyme@posgrado.upv.es, {jsanchis,xblasco,mmiranzo}@isa.upv.es

Abstract. Controller design has evolved to a multiobjective task, *i.e.*, today is necessary to take into account, besides any performance requirement, robustness requisites, frequency domain specifications and uncertain model parameters in the design process. The designer (control engineer), as Decision Maker, has to select the best choice according to his preferences and the trade-off he wants to achieve between conflicting objectives. In this work, a new multiobjective optimization approach using Differential Evolution (DE) algorithm is presented for the design of (but not limited to) Laplace domain controllers. The methodology is used to propose a set of solutions for an engineering control benchmark, all of them non-dominated and pareto-optimal. The obtained results shows the viability of this approach to give a higher degree of flexibility to the control engineer at the decision making stage.

1 Introduction

Nowadays, design techniques in engineering control have evolved. Controller design techniques oriented to fulfill a single specification are not suitable because industry needs controllers that fulfil simultaneously several requirements and performance specifications.

Multiobjective optimization can deal with problems with multiple objectives, since all are important to the designer. It differs from single objective optimization in considering all the objectives simultaneously without a weighting vector or any *a priori* representation of the designer´s preferences.

In this work, a multiobjective optimization algorithm is presented for controller design with several performance objectives and robustness requirements. Additionally, a multiobjective methodology is proposed which includes the identification and control design steps. The algorithm and methodology are validated with an engineering control benchmark.

The rest of this work remains as follows: in section 2 the multiobjective optimization statement is presented; in section 3 the multiobjective optimization tool used in this work is proposed and explained. In section 4 the benchmark example is discussed. Finally, in section 5 some concluding remarks are given.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 532–541, 2010.

2 Multiobjective Optimization Statement

The multiobjective optimization problem, without loss of generality, can be stated as follows:

$$\min_{\boldsymbol{\theta}\in\Re^n} \boldsymbol{J}(\boldsymbol{\theta}) = [J_1(\boldsymbol{\theta}), \ldots, J_m(\boldsymbol{\theta})] \in \Re^m \tag{1}$$

where $\boldsymbol{\theta} \in \Re^n$ is defined as the decision vector and $\mathbf{J}$ as the objective vector. In general, it does not exists a unique solution because there is not a solution better than other in all the objectives. Then a set of solutions, the Pareto set $\boldsymbol{\Theta_P}$ is defined and its projection into the objective space $\mathbf{J_P}$ is known as the Pareto front. Each point in the Pareto front is said to be a non-dominated solution (see definition 1).

Definition 1. *Dominance: given a solution* $\boldsymbol{\theta^1}$ *with cost function value* $\boldsymbol{J(\theta^1)}$ *dominates a second solution* $\boldsymbol{\theta^2}$ *with cost value* $\boldsymbol{J(\theta^2)}$ *if and only if:*

$$\left\{\forall i \in [1,2,\ldots m], J_i(\boldsymbol{\theta^1}) \leq J_i(\boldsymbol{\theta^2})\right\} \wedge \left\{\exists q \in [1,2,\ldots m] : J_q(\boldsymbol{\theta^1}) < J_q(\boldsymbol{\theta^2})\right\}$$

which is denoted as $\boldsymbol{\theta^1} \prec \boldsymbol{\theta^2}$

Multiobjective optimization techniques search for the best discrete approximation $\mathbf{\Theta_P^*}$ of the Pareto set which generates the best description for the Pareto front $\mathbf{J_P^*}$. In this way, the Decision Maker has a set of solutions for a given problem and a higher degree of flexibility to choose a particular or desired solution.

3 Multiobjective Optimization Tool: sp-MODE

3.1 Multiobjective Optimization Algorithm Design

The two main challenges for a multiobjective optimization algorithm are:

- To avoid premature convergence into suboptimal solutions, losing description and generalization on the true Pareto front $\mathbf{J_P}$.
- To find a set of solutions with enough diversity in order to have a good discrete representation of the Pareto front $\mathbf{J_P}$.

A multiobjective optimization algorithm must deal with these challenges in order to offer to the designer a good-quality set of solutions, descriptive enough to allow flexible decisions. To overcome these difficulties a multiobjective optimization algorithm, the sp-MODE has been developed. It is an evolutionary algorithm based on the Differential Evolution (DE) algorithm [1]. This algorithm, instead using an ϵ-dominance concept [2] or related approaches [3,4] to get well-spread solutions and promote diversity, it uses an Spherical Pruning (definition 6) in the objective space to reduce the cardinality of Θ_P^*. Spherical Selection Pruning helps to overcome the drawbacks of ϵ-dominance related methods, where non-dominated solutions could be lost (see figure 1) in the evolution process [4]. As other multiobjective optimization algorithms, it uses an initial population $P(0)$ to explore the search space, it stores the best solutions found so far into a file A and uses them in the evolution process.

Definition 2. ***Spherical Coordinates:*** *given a solution* $\boldsymbol{\theta^1}$ *and* $\mathbf{J^1} = \mathbf{J}(\boldsymbol{\theta^1})$*, let it be* $S(\mathbf{J^1}) = [N_2(\mathbf{J^1}), \boldsymbol{\alpha}(\boldsymbol{J^1})]$ *its normalized hyperspherical coordinates from the ideal solution* $J^{ideal} = \left[\min_{J^i \in J_P^*} J_1^i, \ldots, \min_{J^i \in J_P^*} J_m^i\right]$*, where* $\boldsymbol{\alpha}(\boldsymbol{J^1}) = [\alpha_1^1, \ldots, \alpha_{m-1}^1]$ *is the arc vector and* $N_2(\mathbf{J^1})$ *the Euclidean distance to the normalized ideal solution.*

Definition 3. ***Sight Range:*** *The sight range from the ideal solution to the Pareto front approximation* J_P^* *is bounded by* $\boldsymbol{\alpha^U}$ *and* $\boldsymbol{\alpha^L}$*:*

$$\boldsymbol{\alpha^U} = \left[\max_{J^i \in J_P^*} \left(\alpha_1^{\mathbf{J^i}}\right), \ldots, \max_{J^i \in J_P^*} \left(\alpha_{m-1}^{\mathbf{J^i}}\right)\right], \quad \boldsymbol{\alpha^L} = \left[\min_{J^i \in J_P^*} \left(\alpha_1^{\mathbf{J^i}}\right), \ldots, \min_{J^i \in J_P^*} \left(\alpha_{m-1}^{\mathbf{J^i}}\right)\right]$$

Definition 4. ***HyperCone Grid:*** *Given a set of solutions in the objective space, the hypercone grid on the m-dimensional space in arc increments* $\boldsymbol{\alpha_\epsilon} = [\alpha_1^\epsilon, \ldots, \alpha_{m-1}^\epsilon]$ *is defined as:* $\boldsymbol{\Lambda}^{\mathbf{J_P^*}} = \frac{\boldsymbol{\alpha^U} - \boldsymbol{\alpha^L}}{\boldsymbol{\alpha_\epsilon}} = \left[\frac{\alpha_1^U - \alpha_1^L}{\alpha_1^\epsilon}, \ldots, \frac{\alpha_{m-1}^U - \alpha_{m-1}^L}{\alpha_{m-1}^\epsilon}\right]$

Definition 5. ***Spherical Sector:*** *The normalized spherical sector of a solution* θ^1 *is defined as:* $\boldsymbol{\Lambda}_\epsilon(\boldsymbol{\theta^1}) = \left[\left[\frac{\alpha_1^1}{\Lambda_1^{J_P^*}}\right], \ldots, \left[\frac{\alpha_{m-1}^1}{\Lambda_{m-1}^{J_P^*}}\right]\right]$

Definition 6. ***Spherical Pruning:*** *given two solutions* $\boldsymbol{\theta^1}$ *and* $\boldsymbol{\theta^2}$*,* $\boldsymbol{\theta^1}$ *has an spherical preference over* $\boldsymbol{\theta^2}$ *if and only if:*

$$\left[\boldsymbol{\Lambda_\epsilon}(\boldsymbol{\theta^1}) = \boldsymbol{\Lambda_\epsilon}(\boldsymbol{\theta^2})\right] \wedge \left[N_2(\boldsymbol{J^1}) < N_2(\boldsymbol{J^2})\right]$$

With this new concept, the multiobjective optimization algorithm is less sensitive to lose non-nominated solutions due to the *a priori* unknown geometry of the Pareto front. This approach can be understood as, if the designer stands at the ideal solution (or any desired solution), he will be searching for the nearest non-dominated solution in the Pareto front in any possible direction using discrete arc increments (*vid.* figure 1).

3.2 Multiobjective sp-MODE Algorithm

Due to the multiobjective nature of the optimization approach, just the DE standard mutation and crossover operators will be used in the evolution mechanism of the sp-MODE algorithm.

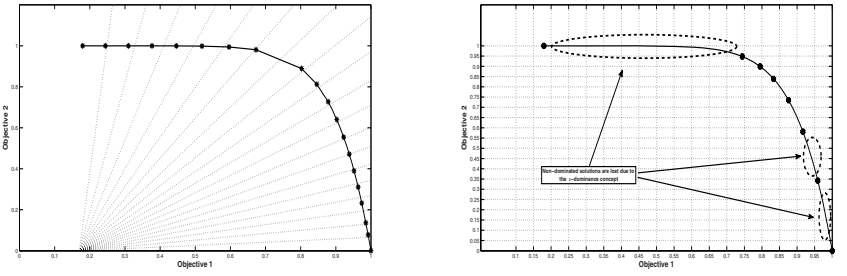
Mutation: For each target (parent) vector $\mathbf{x}_{i,k}$, a mutant vector $\mathbf{v}_{i,k}$ is generated at generation k according to equation 2:

$$\mathbf{v}_{i,k} = \mathbf{x}_{r_1,k} + F(\mathbf{x}_{r_2,k} - \mathbf{x}_{r_3,k}) \tag{2}$$

Where $r_1 \neq r_2 \neq r_3 \neq i$ and F is known as the Scaling Factor.

Crossover: For each target vector $\mathbf{x}_{i,k}$ and its mutant vector $\mathbf{v}_{i,k}$, a trial (child) vector $\mathbf{u}_{i,k} = [u_{i,1,k}, u_{i,2,k}, \ldots, u_{i,m,k}]$ is created as follows:

$$u_{i,j,k} = \begin{cases} v_{i,j,k} & if\ rand(0,1) \leq Cr \\ x_{i,j,k} & otherwise \end{cases} \tag{3}$$



(a) (b)

Fig. 1. Spherical pruning and box selection comparison for two objectives. While ϵ-dominance criterion uses hyperboxes to get well spread solutions (b), the Spherical Selection Pruning uses hypercones to get well spread solutions (a).

where $j \in 1, 2, 3 \ldots n$ and Cr is the Crossover probability rate.

More details about these operators and implementation using DE can be found in [1,5]. The main steps of the multiobjective optimization sp-MODE algorithm are as follows:

Step 1: Initialize $P(0)$ with N_p individuals randomly selected from the decision space $\theta \in \Re^n$.
Step 2: Evaluate initial population $P(0)$.
Step 3: Look for non-dominated solutions on $P(0)$ (definition 1) to get $D(0)$.
Step 4: Perform spherical pruning (definition 6) in $D(0)$ to get $A(0)$ and store the solutions.
Step 5: For $k = 1 :$ **MaxGen** and convergence criterion not reached
  Step 5.1: Select randomly from $P(k)$ and $A(k)$ a subpopulation $N_S(k)$.
  Step 5.2: Apply the fundamental DE operators on subpopulation $N_S(k)$ to get the offspring $O(k)$:
    1. Mutation (equation 2)
    2. Crossover (equation 3)
  Step 5.3: Evaluate offspring $O(k)$; if child $\prec$ parent, the parent is substituted by its child.
  Step 5.4: Apply dominance on $A(k) \bigcup O(k)$ to get $D(k)$
  Step 5.5: Apply Spherical Pruning on $D(k)$ to get $A(k+1)$
  Step 5.6: Store the solution $A(k+1)$
Step 6: Algorithm terminates. Solutions in A are the approximations to the Pareto set and Pareto front, $\mathbf{\Theta}^*_{\mathbf{P}}$ and $\mathbf{J}^*_{\mathbf{P}}$, respectively.

To validate the sp-MODE algorithm, a subset of the IEEE CEC2007 benchmark problems on non-constrained multiobjective optimization [6] were used. The benchmark problems selected were WFG1, WFG8 and WFG9 for $m = 3$ and $m = 5$ objectives.

Table 1. The results for hypervolume indicator $I_{\overline{H}}$ (left) and R indicator (right) on test problems for 5e+5 function evaluations

m		WFG1	WFG8	WFG9
3	Best	6.62E-02	-2.16E-01	-1.22E-01
	Median	1.74E-01	-1.86E-01	-1.02E-01
	Worst	1.97E-01	-1.55E-01	-8.24E-02
	Mean	1.59E-01	**-1.88E-01**	**-1.02E-01**
	Std	4.45E-02	2.90E-02	1.64E-02
5	Best	8.60E-02	-3.91E-01	-2.04E-01
	Median	1.80E-01	-3.22E-01	-1.68E-01
	Worst	2.22E-01	-2.89E-01	-1.34E-01
	Mean	1.62E-01	**-3.31E-01**	**-1.68E-01**
	Std	4.45E-02	2.90E-02	1.64E-02

m		WFG1	WFG8	WFG9
3	Best	1.10E-02	-2.82E-02	-1.51E-02
	Median	3.29E-02	-2.27E-02	-1.17E-02
	Worst	3.71E-02	-1.72E-02	-8.35E-03
	Mean	2.99E-02	**-2.35E-02**	**-1.18E-02**
	Std	6.99E-03	2.54E-03	1.72E-03
5	Best	5.35E-03	-1.08E-02	-1.76E-03
	Median	1.31E-02	-7.68E-03	1.89E-04
	Worst	1.63E-02	-4.46E-03	1.92E-03
	Mean	1.16E-02	**-7.66E-03**	1.82E-04
	Std	3.55E-03	1.63E-03	9.77E-04

The sp-MODE was run with the following parameter values: scaling factor $F = 0.5$, crossover rate $Cr = 0.1$, initial population size $car(P(k)) = 100$, subpopulation size $car(N_S(k)) = 100$, arc increments $\alpha_\epsilon = [16, 16]$ for problems with three objectives and $\alpha_\epsilon = [6, 6, 6, 6]$ with 5 objectives. Scaling factor, crossover rate and population size are adjusted as reported by the GDE3 algorithm [7] for comparison purposes. The GDE3 is a multiobjective optimization algorithm which uses Differential Evolution and among the algorithms which share this characteristic in the CEC2007 benchmark (MOSaDE [8], DEMOwSA [9], MO_DE [10]) was the one with the the best ranking. As reported by GDE3, these fixed parameters are used instead of parameter adjustment to show how the algorithm contributes to the results.

The sp-MODE were tested under the conditions imposed by the CEC2007 benchmark and as result two indicators were obtained. By one hand the R indicator is a performance index of the J_P^* utility compared with a reference set. On the other hand the hypervolume difference to a reference set ($I_{\overline{H}}$) measures the quality of J_P^* to cover the objective space. In both cases, the smaller the value, the better the Pareto front approximation. Results are shown in table 1.

On $m = 3$ problems, the sp-MODE has the second best value for hypervolume on problems WFG1 and WFG8 whilst has the better value on problem WFG9. Concerning the R indicator, the sp-MODE achieves a lesser performance on problems WFG1 and WFG8, but on problem WFG9 has the second best value. Referring to $m = 5$ problems, the sp-MODE is among the first three places for both indicators.

To test the effectiveness and applicability of the algorithm, a solution to an engineering control benchmark (ECB) example is proposed in the following section.

4 Application Example: Helicopter Pitch Angle Control

4.1 Benchmark Description

The multiobjective approach will be used to propose a set of solutions to the contest organized by the CEA-IFAC 2007 Control Engineering group [11]. The engineering control benchmark (ECB07) consists in the design of a controller $G_k(s)$ for the pitch angle of an academic helicopter bench (figure 2). The ECB07 transfer function for the pitch angle α (in radians) and voltage ν (in volts)

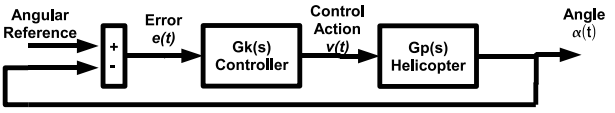


Fig. 2. Control Loop for the ECB07. The objective of the benchmark is to design the controller $G_k(s)$ to regulate the pitch angle of the helicopter $G_p(s)$.

was stated by the organizers (equation 4) with the uncertain parameters $k \in [0.07, 0.12]$, $\xi \in [0.10, 0.16]$, $\omega_n \in [0.55, 0.60]$ and $T \in [0.09, 0.11]$.

A Matlab/Simulink© file with the model and a script to evaluate the controller performance under a known reference trajectory was available to the participants. The benchmark ended with an evaluation on the real helicopter with the given trajectory, but in this work just simulation studies are presented. Details on results and winning controllers can be found in [12].

$$Gp(s) = \frac{\alpha(s)}{\nu(s)} = \frac{k\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} e^{-sT} \tag{4}$$

In order to select a nominal model for control design purposes, a multiobjective optimization approach were followed as proposed in [13]. The sp-MODE was run with an initial population of 100 solution vectors and a subpopulation size of 16 individuals to find the Pareto Front in 200 generations ($F = 0.5$, $Cr = 0.9$, $\alpha_\epsilon = [6, 6, 6]$). Simulations were carried out on a Windows XP PC, Pentium(R) 4 processor, 3.4 Ghz and 1.96 Gb Ram. The optimization process took 0.5 hours. As a result, a set of 425 different models were found.

The model with the best compromise between objectives and confidence indexes, *i.e.* the solution vector with lower N_∞ value in the normalized objective space. This leads to the model of equation 5.

$$\hat{G}_p = \frac{\alpha(s)}{\nu(s)} = \frac{0.114 \cdot 0.587^2}{s^2 + 2 \cdot 0.156 \cdot 0.587 s + 0.587^2} e^{-0.099s} \tag{5}$$

4.2 Controller Design

The multiobjective approach could be used to design any kind of controller (see for example [13]), but for comparison purposes, Laplace domain controllers will be designed. A continuous PID controller based on the structure of the nominal controller defined by the organizers (equation 6) will be proposed and its performance will be compared with the winning controller $G_{w1}(s)$ and the second classified $G_{w2}(s)$ at the ECB07.

$$G_k(s) = \frac{x_1 (s + x_2)(s + x_3)}{s (s + x_4)} \tag{6}$$

Search and Objective Space Definition. The following objectives are defined:

J_1 : Multiplicative Error $\left\| W \frac{G_k \hat{G_p}}{1+G_k \hat{G_p}} \right\|_\infty$

J_2 : Additive Error $\left\| \frac{G_k}{1+G_k \hat{G_p}} \right\|_\infty$

J_3 : Time weighting absolute error on ramp reference signal. $J_3 = \int_{30}^{50} |e(t)| \cdot t \cdot dt$

J_4 : Time weighting absolute error in presence of a disturbance. $J_4 = \int_{50}^{80} |e(t)| \cdot t \cdot dt$

J_5 : Time weighting absolute error on a step reference. $J_5 = \int_{80}^{95} |e(t)| \cdot t \cdot dt$

J_6 : Time weighting absolute error on a parabolic reference. $J_6 = \int_{95}^{120} |e(t)| \cdot t \cdot dt$

J_7 : Integral of control action $J_7 = \int_{30}^{120} |\nu(t)| \cdot dt$

J_8 : Integral of the derivative of the control action $J_8 = \int_{30}^{120} \left| \frac{\nu(t)}{dt} \right| \cdot dt$

Where $e(t)$ is the error and $\nu(t)$ is the control action (figure 2). Objectives J_3 to J_8 are defined by the benchmark organizers with a specific weighting vector β [11]. The ECB07 main objective is to minimize $J_{ECB07} = \beta \cdot [J_3 \dots J_8]^T$ on the real model. Objectives J_1 and J_2 are included in order to assure robustness. W is the usual bound for the parametric uncertainty, and it will be computed using the remaining models of the Pareto front (for details see [14]).

The bounds defined for the search space are: $x_1 \in [0, 10000]$, $x_2 \in [0, 50]$, $x_3 \in [0, 50]$ and $x_4 \in [0, 100]$. The parameter settings are $F = 0.5$, $Cr = 0.9$, $\alpha_\epsilon = [6, 6, 6, 6, 6, 6, 6, 6]$. In a first sp-MODE run, a set of 800 optimal controllers in 500 generations were found.

To visualize the decision and objective spaces, the Level Diagram Tool is used [15]. Level Diagram Tool is based on the classification of the Pareto front obtained where every objective is normalized with respect to its minimum and maximum values. Then, a norm is applied to evaluate the distance to the ideal point $\boldsymbol{J_{ideal}} = [\min(J_1), \dots, \min(J_m)]$.

The level diagram lets the designer include visualization preferences discriminating non desired solutions. In this case, the ECB07 is very specific on its evaluation criteria: the organizers have preferences on controllers with $J_{ECB07} < 1$ ($J_{ECB07} = 1$ is the base controller performance provided). Analyzing the first solution, the controllers which satisfy this requirement lie in $x_1 \in [1000, 5000]$, $x_2 \in [1, 4]$, $x_3 \in [0, 0.1]$ and $x_4 \in [10, 30]$ approximately. Therefore, a new multiobjective run were carried and a total of 427 new controllers on the specified subspace are found in 500 generations. The total time of computation for both runs were approximately 3 hours. The Level Diagram of the last solution obtained with the sp-MODE algorithm is shown on figure 3.

The following geometrical remarks (left) and specific control remarks (right) can be noticed:

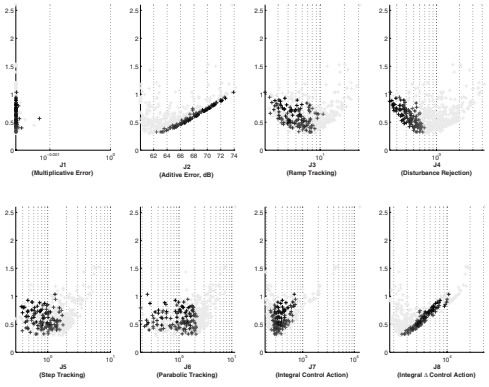


Fig. 3. Pareto front for controller G_k coloured according to the ECB07 preferences ($J_{ECB07} < 1$). The darker the solution, the lower J_{ECB07}. Light solutions do not match the ECB07 preferences ($J_{ECB07} > 1$).

1: The lower J_3 or J_4, the farther from the solution with minimum N_2.
2: The higher J_2, J_7 and J_8, the farther from the solution with minimum N_2.
3: Objectives J_5 and J_6 are well spread into the objective space.

4: Objectives J_3 and J_4 are in conflict with objectives J_2, J_7 and J_8.
5: J_4 has a positive correlation with J_{ECB07}. (The darker the solution, the lower J_4).

1,2: The better the ramp tracking (J_3) or the better the disturbance rejection (J_4), the higher the noise sensitivity (J_2) for control actions.

3: The step tracking (J_5) and parabolic tracking (J_6) are not determinant to achieve $J_{ECB07} < 1$.

4: Performance (J_3,J_4) is in conflict with smooth control actions ($J2$, J_7, J_8), and the designer must found a satisfactory trade-off.
5: Disturbance rejection (J_4) is determinant to achieve $J_{ECB07} < 1$.

A controller with the lowest high frequency gain and the best possible J_{ECB07} performance is needed. On the one hand, a controller with low high frequency gain has a low noise sensitivity and on the other hand a controller with the best performance possible would lead to a less robust controller (typical trade-off between performance and robustness). Using the Level Diagram tool, is possible to identify controllers matching this requirement.

The controller $G_k(s) = \frac{1736.1211(s+1.9234)(s+0.0017)}{s(s+10.4978)}$ has been selected from the Pareto front approximation $\mathbf{J}^*_{\mathbf{P}}$ due to its good disturbance performance and its high frequencies maximum gain indexes. For validation, the winning controllers $G_{w1}(s)$ and $G_{w2}(s)$ (see [12] for details) and the proposed one were tested on

Table 2. J_{ECB07} performance of the proposed controller (G_k) and winning controllers of the ECB07 (G_{w1},G_{w2}) on 50 random plants

	Noise ±0			Noise ±0.0015/2			Noise ±0.0015		
	G_{w1}	G_{w2}	G_k	G_{w1}	G_{w2}	G_k	G_{w1}	G_{w2}	G_k
Avg	0.719	**0.532**	0.651	0.780	0.921	**0.734**	0.871	1.429	**0.868**
Des	0.112	**0.008**	0.051	0.094	**0.018**	0.064	0.076	**0.013**	0.068

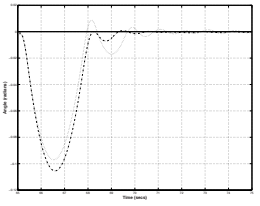


Fig. 4. J_{ECB07} performance assessment in the presence of a disturbance of the proposed controller (...) and the winning controller (-.-) with noise on measurements

50 random plants with different noise levels on measurements (±0, ±0.0015/2 and ±0.0015 radians). The results are given in table 2. In general the proposed controller $G_k(s)$ has a slight improvement over the winning controller $G_{w1}(s)$ respect its performance, having a reduced order. Consequently, it is easier to implement. Controller $G_{w2}(s)$ shows better performance on the first test, but it is quickly degraded by noise.

5 Conclusions

A new multiobjective optimization algorithm, the sp-MODE has been presented. This algorithm shows competitive performances among other MODE algorithms. With the sp-MODE, a set of Laplace domain controllers has been designed to match the specifications defined by the ECB07. The multiobjective optimization lets us define a set of controllers, all of them non-dominated, with different trade-offs in the objective space.

During the decision making phase, the control engineer has a higher degree of flexibility to choose a particular or desired solution. Therefore, it was possible to select a controller according to the EBC07 preferences with a lower order than the winning controller and with a better performance on the simulations performed.

Acknowledgments. This work is partially funded by DPI2008-02133/DPI, Ministerio de Ciencia e Innovación, Gobierno de España.

References

1. 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)
2. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary Computation (3), 263–282 (2002)
3. Herrero, J.M., Martínez, M., Sanchis, J., Blasco, X.: Well-distributed pareto front by using the epsilon-moga evolutionary algorithm. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 292–299. Springer, Heidelberg (2007)
4. Hernández-Días, A.G., Santana-Quintero, L.V., Coello Coello, C.A., Molina, J.: Pareto-adaptive ϵ-dominance. Evolutionary Computation 15(4), 493–517 (2007)
5. Storn, R.: Sci: Differential evolution research: Trends and open questions. In: Chakraborty, U.K. (ed.) Advances in Differential Evolution. SCI, vol. 143, pp. 1–31. Springer, Heidelberg (2008)
6. Huang, V.L., Qin, A.K., Deb, K., Zitzler, E., Suganthan, P.N., Liang, J.J., Preuss, M., Huband, S.: Problem definitions for performance assessment on multi-objective optimization algorithms. Technical report, Nanyang Technological University, Singapore (2007)
7. Kukkonen, S., Lampinen, J.: Performance assesment of generalized differential evolution 3 (gd3) with a given set of problems. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007), September 2007, pp. 3593–3600 (2007)
8. Huang, V.L., Qin, A.K., Suganthan, P.N., Tasgetiren, M.F.: Multi-objective optimization based on self-adaptive differential evolution algorithm. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007), September 2007, pp. 3601–3608 (2007)
9. Zamuda, A., Brest, J., Boskovic, B., Zumer, V.: Differential evolution for multiobjective optimization with self adaptation. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007), September 2007, pp. 3617–3624 (2007)
10. Zielinski, K., Laur, R.: Differential evolution with adaptive parameter setting for multi-objective optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007), September 2007, pp. 3585–3592 (2007)
11. García-Sanz, M., Elso, J.: Ampliación del benchmark de diseño de controladores para el cabeceo de un helicóptero. Revista Iberoamericana de Automática e Informática Industrial 4(1), 107–110 (2007)
12. García-Sanz, M., Elso, J.: Resultados del benchmark de dise no de controladores para el cabeceo de un helicóptero. Revista Iberoamericana de Automática e Informática Industrial 4(4), 117–120 (2007)
13. Reynoso-Meza, G., Blasco, X., Sanchis, J.: Diseño multiobjetivo de controladores pid para el benchmark de control 2008-2009. Revista Iberoamericana de Automática e Informática Industrial 6(4), 93–103 (2009)
14. Gu, D.W., Petkov, P.H., Konstantinov, M.M.: Robust control design with Matlab. Springer, Heidelberg (2005)
15. Blasco, X., Herrero, J.M., Sanchis, J., Martínez, M.: A new graphical visualization of n-dimensional pareto front for decision-making in multiobjective optimization. Information Sciences 178(20), 3908–3924 (2008)

Parameter Tuning of Evolutionary Algorithms: Generalist vs. Specialist

S.K. Smit and A.E. Eiben

Vrije Universiteit Amsterdam
The Netherlands
{sksmit,gusz}@cs.vu.nl
http://mobat.sourceforge.net

Abstract. Finding appropriate parameter values for Evolutionary Algorithms (EAs) is one of the persistent challenges of Evolutionary Computing. In recent publications we showed how the REVAC (Relevance Estimation and VAlue Calibration) method is capable to find good EA parameter values for single problems. Here we demonstrate that REVAC can also tune an EA to a set of problems (a whole test suite). Hereby we obtain robust, rather than problem-tailored, parameter values and an EA that is a 'generalist, rather than a 'specialist. The optimized parameter values prove to be different from problem to problem and also different from the values of the generalist. Furthermore, we compare the robust parameter values optimized by REVAC with the supposedly robust conventional values and see great differences. This suggests that traditional settings might be far from optimal, even if they are meant to be robust.

Keywords: parameter tuning, algorithm design, test suites, robustness.

1 Background and Objectives

Finding appropriate parameter values for evolutionary algorithms (EA) is one of the persisting grand challenges of the evolutionary computing (EC) field. As explained by Eiben *et al.* in [8] this challenge can be addressed before the run of the given EA (parameter tuning) or during the run (parameter control). In this paper we focus on parameter tuning, that is, we are seeking good parameter values off-line and use these values for the whole EA run. In today's practice, this tuning problem is usually 'solved' by conventions (mutation rate should be low), ad hoc choices (why not use uniform crossover), and experimental comparisons on a limited scale (testing combinations of three different crossover rates and three different mutation rates). Until recently, there were not many workable alternatives. However, by the developments over last couple of years now there are a number of tuning methods and corresponding software packages that enable EA practitioners to perform tuning without much effort. In particular, REVAC [10,13] and SPOT [3,5,4] are well developed and documented.

The main objective of this paper is to illustrate the advantage of using tuning algorithms in terms of improved EA performance. To this end, we will select

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 542–551, 2010.

a set of test functions and compare a benchmark EA (with robust parameter values set by 'common wisdom') with an EA whose parameters are tuned for this set of functions. A second objective is to compare specialist EAs (that are tuned on one of our test functions) with a generalist EA (that is tuned on the whole set of test functions). For this comparison we will look at the performance of the EAs as well as the tuned parameter values. Furthermore, we want to show what kind of problems rise when tuning evolutionary algorithms. At the end we hope to provide a convincing showcase justifying the use of a tuning algorithm and to obtain novel insights regarding the good parameter values.

2 Parameters, Tuners, and Utility Landscapes

To obtain a detailed view on parameter tuning we distinguish three layers: the application layer, the algorithm layer, and the design or tuning layer. The lower part of this three-tier hierarchy consists of a problem on the application layer (e.g., the traveling salesman problem) and an EA (e.g., a genetic algorithm) on the algorithm layer trying to find an optimal solution for this problem. Simply put, the EA is iteratively generating candidate solutions (e.g., permutations of city names) seeking one with maximal fitness. The upper part of the hierarchy contains a tuning method that is trying to find optimal parameter values for the EA on the algorithm layer. Similarly to the lower part, the tuning method is iteratively generating parameter vectors seeking one with maximal quality, where the quality of a given parameter vector is based on the performance of the EA using the values of it. To avoid confusion we use the term *utility*, rather than fitness, to denote the quality of parameter vectors. Table 1 provides a quick overview of the related vocabulary.

Using this nomenclature we can define the *utility landscape* as an abstract landscape where the locations are the parameter vectors of an EA and the height reflects utility, based on any appropriate notion of EA performance. It is obvious that fitness landscapes –commonly used in EC– have a lot in common with utility landscapes as introduced here. To be specific, in both cases we have a search space (candidate solutions vs. parameter vectors), a quality measure (fitness vs. utility) that is conceptualized as 'height', and a method to assess the quality of a point in the search space (evaluation vs. testing). Finally, we have a search method (an evolutionary algorithm vs. a tuning procedure) that is seeking for a point with maximum height.

Table 1.

	problem solving	parameter tuning
Method at work	evolutionary algorithm	tuning procedure
Search space	solution vectors	parameter vectors
Quality	fitness	utility
Assessment	evaluation	testing

3 Generalist EAs vs. Specialist EAs

Studying algorithm performance on different problems has led to the no-free-lunch theorem stating that algorithms (of a certain generic type) performing well on one type of problem, will perform worse on another [14]. This also holds for parameter values in the sense that a parameter vector that performs good on one type of problems, is likely to perform worse on another. However, very little effort is spent on studying the relation between problem characteristics and optimal parameter values. It might be argued that this situation is not a matter of ignorance, but a consequence of an attitude favoring robust parameter values that perform good on a wide range of different problems. Note that the term 'robust' is also used in the literature to indicate a low variance in outcomes when performing multiple repetitions of a run on the same problem with different random seeds. To avoid confusion, we will use the term *generalist* to denote parameter values that perform good on a wide range of problems. The opposite of such a generalist is then a *specialist*, namely a parameter set that shows excellent performance on one specific type of problems.

The notion of a generalist raises a number of issues. First, a true generalist would perform good on all possible test functions. However, this is impossible by the no-free-lunch theorem. So, in practice, one needs to restrict the set of test functions a generalist must solve well and formulate the claims accordingly. For instance, a specific test suite $\{F_1, \ldots, F_n\}$ can be used to support such claims.

The second problem is related to the definition of utility. In simplest case, the utility of a parameter vector p is the performance of the EA using the values of p *on a given test function* F. This notion is sufficient to find specialists for F. However, for generalists, a collection of functions $\{F_1, \ldots, F_n\}$ should be used. This means that the utility is not a single number, but a vector of utilities corresponding to each of the test functions. Hence, finding a good generalist is a multi-objective problem, for which each test-function is one objective. In this investigation we address this issue in a straightforward way, by defining the utility on a set $\{F_1, \ldots, F_n\}$ as the average of utilities on the functions F_i.

4 Experimental Setup and System Description

As described earlier, the experimental setup consist of a three layer architecture. On the **application layer**, we have chosen a widely used set of 10 dimensional test-functions to be solved, namely: Ackley, Griewank, Sphere, Rastrigin, and Rosenbrock. For Ackley, Griewank and Rosenbrock, the Evolutionary Algorithm is allowed for 12.000 function evaluations. On Rastrigin it is allowed for 10.000 evaluations, and on the Sphere function only 8.000. This is a rather limited set of problems, but due to a large runtime more exhaustive and complex test suites are not yet feasible.

On the **algorithm layer**, we have chosen a simple genetic algorithm using N-point crossover, bitflip mutation, k-tournament parent selection, and deterministic survivor selection. These choices require 6 parameters to be defined as

Table 2. Parameters to be tuned, and their ranges

Parameter	Min	Max
Population size	2	200
Offspring size	1	200
Mutation probability	0	1
# crossover points	1	149
Crossover probability	0	1
Tournament size	1	200

described in Table 2. the allowed values for most of the parameters a define by either the population size or genome length (150). For population size, we have chosen a maximum value of 200, which we believe is big enough for this genome size and allowed number of evaluations.

The offspring size determines the number of individuals that are born every generation. These newborn individual replace the worst individuals in the population. If the offspring size is bigger than the population size, then the whole population is replaced by the new group of individuals, causing an increase in population size. N-point crossover is chosen to allow for a whole range of crossover operators, such as the commonly used 1-point crossover (N=1) and 2-point crossover (N=2). The same holds for k-tournament selection. The commonly used random uniform selection (k=1), deterministic selection (k $\geq$ population-size) and everything in between can be used by means of selecting k accordingly. Because the test-functions require 10 dimensional real-valued strings as input, a 15-bit Gray coding is used to transform the binary string of length 150, into a real-valued string of length 10.

On the **design layer**, REVAC [9] is used for tuning the parameters of the Evolutionary Algorithm. Technically, REVAC is a heuristic generate-and-test method that is iteratively searching for the set of parameter vectors of a given EA with a maximum performance. In each iteration a new parameter vector is generated and its performance is tested. Testing a parameter vector is done by executing the EA with the given parameter values and measuring the EA performance. EA performance can be defined by any appropriate performance measure and the results will reflect the utility of the parameter vector in question. Because of the stochastic nature of EAs, in general a number of runs is advisable to obtain better statistics. A detailed explanation of REVAC can be found in [13] and [11].

REVAC itself has some parameters too, which need to be specified. The REVAC-parameter values used in these experiments are the default settings, and can be found in Table 3.

4.1 Human Expert

Tuning an algorithm requires a lot of computer power, while some people argue that this is a waste of time. General rules of thumb as a population size of

Table 3. REVAC Parameters

Population Size	80
Best Size	40
Smoothing coefficient	10
Repetitions per vector	10
Maximum number of vectors tested	2500

Table 4. Best Parameter Values and their Mean Best Fitness (to be minimized). Standard deviations shown within brackets.

	Common	**Generalist**	**Specialist**				
			Ackley	**Griewank**	**Sphere**	**Rastrigin**	**Rosenbrock**
Pop. size	100	173	125	12	34	148	107
Offspring size	2	18	67	83	5	9	94
Mutation prob.	0.006	0.0453	0.0405	0.0261	0.0077	0.0301	0.0393
N-point crossover	2	96	115	19	71	18	27
Crossover prob.	1	0.7733	0.4136	0.8153	0.9236	0.7516	0.9762
Tournament size	5	186	30	53	19	104	118
Ackley	1.755 (1.747)	0.127 (0.539)	0.026 (0.013)	0.113 (0.542)	2.005 (2.312)	0.244 (0.753)	0.243 (0.782)
Griewank	0.089 (0.038)	0.059 (0.023)	0.082 (0.040)	0.081 (0.032)	0.083 (0.056)	0.070 (0.037)	0.079 (0.030)
Sphere	0.007 (0.028)	0.021 (0.015)	0.087 (0.056)	0.01 (0.012)	0.000 (0.000)	0.002 (0.003)	0.029 (0.022)
Rastrigin	14.57 (10.48)	6.92 (4.70)	7.28 (3.65)	10.40 (5.97)	16.23 (11.71)	7.60 (4.99)	9.85 (4.91)
Rosenbrock	134.3 (395.0)	64.2 (110.4)	125.2 (129.4)	68.4 (126.5)	151.7 (229.8)	103.8 (195.2)	62.5 (123.4)
Average	30.1328	14.2726	26.5324	15.7996	34.0037	22.3373	14.5420

100 and low mutation probabilities are supposed to perform reasonably well. The question rises how beneficial tuning, and more specific automated tuning, is even to experienced practitioners. For quick assessment of the added value of algorithmic tuning, we have tested the EA using parameter values defined by 'common wisdom' (Table 4).

4.2 Performance Measures

The quality of a certain parameter vector, is measured two times. First, the estimated utility is used to asses a performance to the parameter-vector during the tuning procedure. This estimated utility is calculated by taking the Mean Best Fitness of 10 independent runs using these parameter-values. Secondly, after the tuning procedure is finished, a validated utility is calculated for the 5 parameter vectors with the best estimated utility. This validated utility is based on 25 independent runs instead of 10, and is therefore supposed to be a better estimate of the true utility of the parameter vector.

After this validation step, we define the 'best parameter values' as the parameter vector with the highest validated utility. Furthermore, we tried to indicate good ranges for each of the parameters. Such a range is defined by taking the value of .25 and .75 quantile of the 25 parameter vectors with the highest estimated utility. This removes outliers that are caused by parameters vectors that were 'lucky', and received a estimated utility that is much higher than their true utility.

4.3 System Description

The complete experiment is defined in MOBAT [12] (Meta-Heuristic Optimizer Benchmark and Analysis Toolbox), a toolbox for defining, tuning and evaluating Evolutionary Algorithms on a distributed system. The default package of MOBAT contains all the components for composing the evolutionary algorithm used in these experiments, the test-functions and REVAC. MOBAT is open source and freely available via SourceForge.net.

The experiments are ran on a single 2.93 GHz Intel Core 2 Duo processor with 4GB of memory. A specialist tuning-session took on average 8 hours to finish, while the generalist experiment on our testsuite of 5 test functions finished in 40 hours.

5 Results

5.1 Performance

In this section we present the outcomes of our experiments. As explained in section 3, our goal is to find the best parameter vectors for both specialists and generalists. Furthermore, we stated that a good generalist, is the one with the best average performance. From the results in Table 4 we can immediately see the effect of this choice. The Rosenbrock function appeared to be much harder to solve than the other functions, causing big differences in utility. Therefore the best generalist, was the one that performed very well on the Rosenbrock function, without losing too much performance on the other functions. From Table 5, it is even clear that there is no significant difference in multi-function performance between the three best performing instances on the Rosenbrock function (based on a t-test with $\alpha = 0.1$). Furthermore, we can observe that the generalist is only outperformed by a specialist on the Sphere function. However, focusing more on the sphere function makes hardly any difference in the average performance, due to the small function values on this problem. The parameter values chosen by 'common wisdom' are, except on the Sphere function, significantly outperformed by the other parameter vectors.

When looking at the specialists, we can observe some interesting phenomena. It is apparently very easy to tune parameters in such a way that they are purely specialized on the Sphere function. This specialist is the only one that solves its problem perfectly, but on the downside, it performs very bad on the others functions. The Ackley specialist, on the other hand, does not only perform best on its 'own' function, but also outperforms most others on the Rastrigin function. Interestingly, the Rosenbrock and Griewank specialists show very similar behavior on all functions, however it is remarkable that the Griewank specialist has only an average performance on the function it is tuned to.

Estimated Utility vs. Validated Utility: One of the causes of such suboptimal performance on its 'own' function, is a difference between the estimated utility, that is used during tuning, and the validated utility, as shown in the

Table 5. The specialists/generalist (colums) that show significantly better performance ($\alpha = 0.1$) on a certain problem(rows), based on the results from Table 4

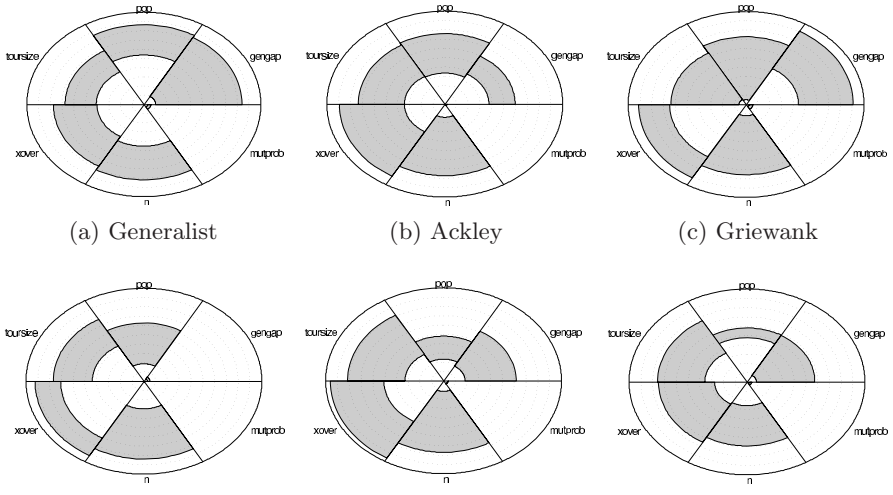
	Common	**Generalist**	**Specialist**				
			Ackley	**Griewank**	**Sphere**	**Rastrigin**	**Rosenbrock**
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Ackley		1,5	1,5,6,7	1,5		1,5	1,5
Griewank		All					
Sphere	2,3,7	3,7		2,3,7	All	1,2,3,4,7	3
Rastrigin		1,4,5,7	1,4,5,7	1,5		1,4,5,7	1,5
Rosenbrock		1,3,5		1,3,5			1,3,5
Average		1,3,5		1,3,5			1,3,5

results. In some cases, these validated utilities are twice as bad as the estimated utility. These differences can be explained by looking at the fitness landscapes of the functions.

In most cases, a fitness landscape consists of multiple local optima and a global optimum, with certain bases of attraction. It is likely that an evolutionary algorithm will terminate with a value that is (close to) a local or global optimum, because it will continue to climb the hill it is currently on, using small mutations until it gets stuck. The utility vector, therefore consists of values that are all close to an optima. For example, the utility vector of the 'common wisdom' parameter values on the Ackley function has 58% of the values close to the global optimum. 35 % of the values is between 3 and 4, which are exactly the values on the first ring of local optima surrounding the global optimum. Finally 7 % of the fitness values at termination, is between 5 and 7, which is equal to value of the second ring of local optima. Such a distribution can disturb the tuning run heavily, for example in 7.5% of the cases the estimated utility, based on 10 runs, of this set of parameter values will be lower than 0.2, which is nine times lower than the 'true' utility. Such a lucky one can therefore steer evolution into the wrong direction.

5.2 Best Parameter Values

To get better insight into the best parameter ranges, we have chosen not to use the single best solution, but the 1% of the best performing parameters values during the tuning phase. Figure 1 shows the .25 and .75 quantile of those values for each of the specialists and the generalist relative to the parameter ranges. The most obvious conclusion that can be drawn from these charts, is that mutation should be low, in order to get a good performance. However, there are also some less standard conclusions. One of the most interesting results is that, although the ranges for population-size are quite different for each of the functions, the 'guesstimate' of population-size equals 100 is not that bad after all. To be more specific, on all five problems, a population-size of 100 lies within the .25 and .75 quantile. However, most other parameter values are completely different than the 'common wisdom' ones. For example the N parameter of N-Point crossover is much higher than 2, which indicated that uniform crossover like crossover,



(a) Generalist (b) Ackley (c) Griewank

(d) Sphere (e) Rastrigin (f) Rosenbrock

Fig. 1. The good parameter ranges for each of the parameters on each test function, and the combined set. The parameter ranges from Table 2 are scaled to $[0, r]$.

outperforms common 1 or 2-point crossover on these problems. This is probably due to the separable (Sphere, Rastrigin and Ackley) or partially separable nature of the test functions.

Furthermore, we can observe a much higher selection pressure than normally used. Tournament sizes are almost equal to the population size, causing the evolutionary algorithm to rapidly converge towards the best individuals in the population. However, such behavior can be explained by the limited number of evaluations that the evolutionary algorithm was allowed to perform. The question rises, if such a fast-converging algorithm is always preferred over a slow-but-accurate instance. In some cases it is, while in other cases it might not be the preferred behavior. Therefore, we emphasize that the 'best parameter values' presented here are highly related to the choices that are made in the experimental setup.

6 Conclusions and Outlook

In this paper we have shown that REVAC is not only capable to find good EA parameter values for a single problem (test function), but also for a set of problems (test functions). The parameter values we found for our generalist, differ greatly from the 'common wisdom' values that are supposed to be robust. The 'optimal' selection pressure for these problems, appears to be much higher than commonly used. Furthermore, a many-point crossover operator outperforms the commonly used 1 -and 2-point crossover on all five problems. On the other hand, a population-size of 100 turned out to be not that bad after all. The scope of these conclusions is limited, we do not advocate them as being the new best

general parameters, because different test-suites, aggregation functions and performance measures will lead to different 'optimal' parameters. The best generalist will therefore always depend upon the choices that are made to define it. Based on the definition of generalist in this paper, our generalist performed quite good on most problems. However, the results on the Sphere function confirm that the no-free-lunch theorem also holds on a parameter vector level.

Furthermore, the experiments revealed some major issues for parameter tuning in general. Estimating the utility has a key role in parameter tuning, both for specialists and generalists. Our experiments revealed how the number of runs per vector can influence the outcome of the experiments. Too many runs lead to high computational costs, while too few lead to an inaccurate estimated utility and therefore inaccurate results. Therefore we advocate the use of racing [2,6,15,13] and sharpening [4,13] to deal with this issue. This, on the one hand sharpens the estimate of the utility, and on the other hand reduces the computational effort.

Tuning for a generalist raises specific problems. In general, it is not clear how a good generalist should be defined. In the area of multi-objective optimization several approaches are known. One approach is to use aggregation methods, like the simple average that we used here. From the results we can observe the effect of such choices; it is more effective to focus on the 'hard' problems that can lead to high deviation in the average utility, rather than searching for a generalist that performs good on all functions. When defining tuning sessions, one have to be aware of the fact that a tuner will optimize on the problem that is defined, rather than the problem they wished to be solved.

Future work can overcome this issue, by using an approach known from multi-objective optimization, namely searching for the Pareto front. Rather than aggregating the results based on choices made beforehand, such an approach allows the researcher to study the effect of a certain design choice afterwards. In [7] such an approach is used to show which parameter values are optimal, when comparing on both algorithm speed and algorithm accuracy at the same time, without specifying weights for those two objective beforehand. This approach can easily be extended to multi-function optimization, which can give insight into the the ranges of parameter-values that are most efficient on a certain problem, a class of problems, or on a whole test suite.

By means of one extensive run, one can identify specialists, class-specialist, or a true generalist without defining those terms beforehand. Based on such studies, one no longer has to rely on 'common wisdom' in order to choose their parameter values wisely but can select one that fits their needs.

References

1. Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, May 18-21. IEEE Press, Los Alamitos (2009)
2. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In: Hybrid Metaheuristics, pp. 108–122 (2007)

3. Bartz-Beielstein, T., Lasarczyk, C.W.G., Preuss, M.: Sequential parameter optimization. In: Corne, D., et al. (eds.) Proceedings of the 2005 IEEE Congress on Evolutionary Computation IEEE Congress on Evolutionary Computation, Edinburgh, UK, vol. 1, pp. 773–780. IEEE Press, Los Alamitos (2005)
4. Bartz-Beielstein, T., Parsopoulos, K.E., Vrahatis, M.N.: Analysis of Particle Swarm Optimization Using Computational Statistics. In: Chalkis (ed.) Proceedings of the International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2004), pp. 34–37 (2004)
5. Bartz-Beielstein, T., Markon, S.: Tuning search algorithms for real-world applications: A regression tree based approach. Technical Report of the Collaborative Research Centre 531 Computational Intelligence CI-172/04, University of Dortmund (March 2004)
6. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B. (ed.) GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
7. Dréo, J.: Using performance fronts for parameter setting of stochastic metaheuristics. In: Rothlauf, F. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 2197–2200. ACM, New York (2009)
8. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
9. Nannen, V., Eiben, A.E.: Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: Veloso, M.M. (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1034–1039 (2007)
10. Nannen, V., Eiben, A.E.: A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms. In: Keijzer, M. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 183–190. Morgan Kaufmann, San Francisco (2006)
11. Nannen, V., Smit, S.K., Eiben, A.E.: Costs and benefits of tuning parameters of evolutionary algorithms. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 528–538. Springer, Heidelberg (2008)
12. Smit, S.K.: MOBAT (2009), http://mobat.sourceforge.net
13. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: [1]
14. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transaction on Evolutionary Computation 1(1), 67–82 (1997)
15. Yuan, B., Gallagher, M.: Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithms, pp. 121–142. Springer, Heidelberg (2007)

Memory Design for Constrained Dynamic Optimization Problems

Hendrik Richter

HTWK Leipzig, Fakultät Elektrotechnik und Informationstechnik
Institut Mess–, Steuerungs– und Regelungstechnik
Postfach 30 11 66, D–04251 Leipzig, Germany
richter@fbeit.htwk-leipzig.de

Abstract. A proposal for a memory design is given that is suitable for solving constrained dynamic optimization problems by an evolutionary algorithm. Based on ideas from abstract memory, two schemes, blending and censoring, are introduced and tested. Using a new benchmark we show in numerical experiments that such a memory can improve solving certain types of constrained dynamic problems.

1 Introduction

Dynamic optimization problems are characterized by fitness landscapes that change their topology during the run–time of the evolutionary algorithm (EA). Hence, instead of finding the optimum as in solving static problems, dynamic optimization means tracking the movement of the optimum. Constraints restrict the allowed search space of the optimization problem as they define which solutions are permissible and which are not. Geometrically, they can be interpreted as (possibly moving) boundaries in the fitness landscape that separate feasible solutions from infeasible ones. Solving such kind of problems recently attracted much interest, see e.g. [8,15] and references cited there.

A key issue in solving dynamic optimization problems is the management of the population's genetic diversity. For addressing genetic diversity, different types of schemes have been proposed, for instance (self–) adaption of the EA's parameters [1], or random diversity enhancement by hyper–mutation [7] or random immigrants [16], or different types of memory [3,9,13,17]. As it has been shown that memory, in particular, seems to be a promising option in solving dynamic optimization problems, our question is if this remains the case for (possibly dynamic) constraints to the problem. However, to the best of our knowledge, so far there have been no reported attempts to use a memory–based approach for constrained dynamic optimization problems. We here employ a memory scheme recently suggested for solving dynamic optimization problems, abstract memory [9,10], and propose modifications to make it fit for constrained problems, namely blending and censoring. Our results show that for certain types of constrained dynamic problems such a scheme can improve the algorithm's performance.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 552–561, 2010.

We next review work related to memory design for constrained dynamic optimization problems. In Sec. 3 dynamic fitness landscapes with constraints are considered, a new class of benchmark problems is introduced, and two standard schemes for solving static constrained optimization problems, penalty functions and repair algorithms, are recalled. In the numerical experiments, presented in Sec. 5, these two schemes are employed with and without the proposed memory to investigate its usefulness. Prior to this, Sec. 4 gives the memory design. We end with summarizing the results and pointing at future problems.

2 Related Work

Work directly related to memory design for constrained dynamic optimization problems exists in two fields: constraint–handling in evolutionary optimization and memory schemes for dynamic optimization problems. As mentioned before, constraints do restrict the permissible search space and divide solutions into feasible and infeasible ones. For the evolutionary search process this means not only the optima need to be found but also it must be ascertained that they belong to the feasible space. So, the following (possibly contradictory) aims need to be pursued in evolutionary search. On the one hand, computational effort should not be wasted in the infeasible region, but the boundary region between feasible and infeasible solutions should also be searched effectively. While the first aim suggests to discard all infeasible solutions, the latter indicates that at least some infeasible solutions should be employed in the search process and furthermore that infeasible solutions may carry information that is important and usable. Nevertheless, it is easy to imagine that optima situated on small feasible "islands" within a vast infeasible "sea" constitute a problem that is very difficult to solve by evolutionary computation. To solve static optimization problems, different methods have been suggested for constraint–handling [5], which can be categorized into four groups: (i.) The use of penalty functions [4,5], (ii.) employing repair algorithms [12], (iii.) approaches maintaining and using infeasible solutions [15] and (iv.) methods reformulating and solving the constrained problem as a multi–objective (un–constrained) optimization problem [4].

The usage of memory is a popular method to address the diversity issue and has shown to be a promising option in solving dynamic optimization problems [3,9,13,17]. The key points in employing memory are to select useful information about the solving process (usually in form of individuals that have distinguished themselves in optimum seeking), storing the information in an effective and organized way, and finally retrieving it in order to boost exploration after a change has happened. The memory schemes suggested range from simply storing best individuals and re–insert them into the population after a change (called direct memory [3,14]), over storing best individuals together with some quantifiers of the state the dynamic problem has (called associated memory [17]) to storing best individuals to build a probabilistic model for the spatial distribution of future solutions of the problem (called abstract memory [9,10]). Further questions addressed in the literature are the organization of the memory (updating rules, memory size etc.) and which individuals should be selected.

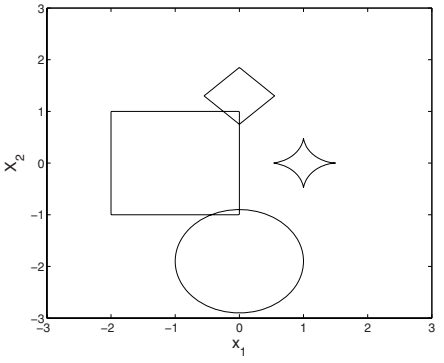
3 Dynamic Fitness Landscapes with Constraints

In dynamic optimization the topological features of the fitness landscape evolve with time, which means that the optima change their coordinates, shapes and surroundings. Constraints define which points in the search space are feasible and which are not. With dynamic constraints this definition changes with the run–time of the EA. This, consequently, has an effect on the dynamic fitness landscape itself, but also on the EA's individuals populating the landscape. For the dynamic fitness landscape with dynamic constraints this may result in the following scenarios. After a change the former global optimum and its surrounding becomes infeasible or there is a connection and/or disconnection between feasible search space regions or the global optimum jumps from one feasible region to another. Apart from the obvious fact that there are feasible and infeasible individuals, for the population a change may also mean that the percentage of infeasible individuals might rise or fall dramatically. All this has consequences for the design of the diversity management needed in dynamic optimization by evolutionary computation.

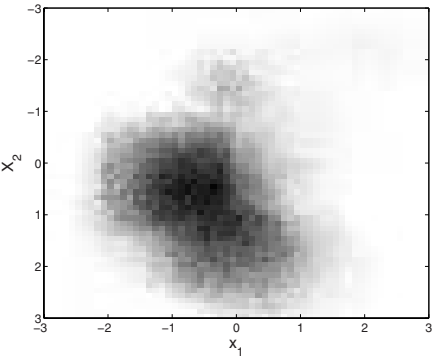
Most known modifications of EAs proposed for solving dynamic optimization problems share the same structural and functional pattern: they (a) detect a change in the dynamic fitness landscape, (b) enhance and/or maintain the population's diversity to enable exploration, (c) try to converge to the optimum with maximal speed by exploitation, and (d) prepare for the next change by controlling and/or maintaining diversity. The main effect of most existing schemes to cope with dynamic environments is modifying the composition of the population after each change in the fitness landscape. As mentioned before this might for instance be accomplished by substantial random alteration of the individuals (hyper–mutation [7]), introducing randomly generated new individuals (random immigrants [16]), retrieving individuals from memory (different kinds of memory schemes [3,9,13,17]) or generating specific individuals on propose (anticipating and predicting the landscape dynamics [2,10,14]). The trouble with all these schemes in constrained dynamic optimization is that the compositional modification might fall flat because all or a considerable part of the "modified population" appears infeasible [8,15]. Of course, the problem could be remedied by checking the feasibility of the modified population, but that would somehow pretend that evaluating feasibility comes at no numerical cost.

Solving constrained dynamic optimization problems is a new field in evolutionary computation and there is a certain lack of established benchmark problems. Recently, a set of problems has been proposed [8]; we here introduce an alternative. It consists of a well–known and often–studied dynamic optimization problem, an n–dimensional "field of cones on a zero plane", where N cones with coordinates $c_i(k)$, $i = 1, \cdots, N$, are moving with discrete time $k \in \mathbb{N}_0$. These cones are distributed across the landscape and have randomly chosen initial coordinates $c_i(0)$, heights h_i, and slopes s_i. So, the dynamic fitness function is

$$f(x,k) = \max\left\{ 0\,,\ \max_{1\le i\le N} \left[h_i - s_i \|x - c_i(k)\|_2\right] \right\}. \tag{1}$$



(a) Norm–based constraints (2)



(b) Image of the memory $\mathcal{C}_\mu$

Fig. 1.

In addition, the dynamic optimization problem is subjected to dynamic norm–based constraints

$$g_j(x,k) = \|b_j x - cc_j(k)\|_{p_j} - r_j \leq 0, \quad j = 1,2,\ldots,m. \tag{2}$$

The exact appearance of the constraints depends on the center point cc_j, the size r_j, the ratio b_j (which defines the relative size with respect to the different spatial directions x) and the shape p_j (which is diamond–like for $p_j = 1$, sphere–like for $p_j = 2$, manhatten–like for $p_j = \infty$, and anything in-between for $0 < p_j < \infty$, see Fig. 1a for an example). An advantage of using norm–based constraints (2) is that the parameters cc_j, r_j, b_j and p_j allow an easy geometrical interpretation. This might be helpful in measuring and evaluating the resulting constrained dynamic fitness landscape. In principle, all these four parameters of the constraints (2) could change with time k. We here only allow the center coordinates $cc_j(k)$ to change with time, assuming that the other parameters are set in initialization and remain constant for the run–time of the EA.

So, the constrained dynamic optimization problem we intend to solve is finding the maximal $x_S \in M$ in a fixed bounded search space $M \subset \mathbb{R}^n$ with $f(x_S,k) \geq f(x,k)$, $\forall x \in M$ for every $k \geq 0$ while at least one of the constraints $g_j(x,k)$ is not violated:

$$f_S(k) = \max_{x \in M} f(x,k), \text{ subject to } (\min_j g_j(x,k)) \leq 0,\ k \geq 0. \tag{3}$$

Due to the analytic nature of $f(x,k)$ and $g_j(x,k)$, the value of $f_S(k)$ can be calculated exactly for every k, which is helpful in determining the EA's performance. The EA we consider for solving problem (3) has a real number representation and λ individuals $p_\kappa \in \mathbb{R}^n$, $\kappa = 1,2,\ldots,\lambda$, which build the population $P \in \mathbb{R}^{n\times\lambda}$. Its dynamics is described by the generation transition function $\psi : \mathbb{R}^{n\times\lambda} \to \mathbb{R}^{n\times\lambda}$, which can be interpreted as a nonlinear probabilistic dynamical system that maps $P(t)$ onto $P(t+1)$ and hence transforms a population at generation $t \in \mathbb{N}_0$ into a population at generation $t+1$, $P(t+1) = \psi(P(t)), t \geq 0$. The time scales t and k are related by the change frequency $\gamma \in \mathbb{N}$ as $t = \gamma k$ with γ being

constant. Further, we assume that the change frequencies and change points for both dynamic fitness function and dynamic constraints are identical.

As pointed out in Sec. 2 there are several ways to deal with (static) constraints. In the experiments reported here, we employ and test two schemes: penalty functions and repair algorithms. Both schemes are briefly recalled next. The working mode of an EA includes calculating the dynamic fitness function $f(x,k)$ and the dynamic constraints $g_j(x,k)$ for each individual p_κ at the beginning of every generation t. For the penalty function method, we calculate for all individuals $p_\kappa(t) \in P(t)$ a penalized fitness function value $f_{pen}(x,k)$ from the dynamic fitness function (1) and the dynamic constraints (2):

$$f_{pen}(p_\kappa(t),k) = f(p_\kappa(t),k) - \sum_{j=1}^{m} \max(0, g_j(p_\kappa(t)),k), \tag{4}$$

where $t = \gamma k$. This quantity is used as the fitness of each individual in the following evolutionary operators (mainly selection, but possibly also recombination and mutation). By the penalized fitness function individuals that are not feasible are punished so that their survival in the evolutionary process becomes less likely. So, it is intended that the population moves towards feasible regions.

For an implementation of a repair algorithm, we use methods from GENOCOP III [6], which works like this. After evaluating the dynamic fitness function $f(x,k)$ and the dynamic constraints $g_j(x,k)$ for all individuals p_κ, the feasible population P_{feas} (defined by its individuals not violating any of the constraints (2)) is separated from the infeasible P_{infeas}. All individuals of P_{infeas} are then repaired by the following procedure. Let p_{infeas} be an infeasible individual from P_{infeas}. We randomly select a feasible individual p_{feas} from P_{feas} and calculate a candidate p_{cand} by $p_{cand} = a\,p_{infeas} + (1-a)\,p_{feas}$ with a being a realization of a uniformly distributed random variable. The process is repeated until p_{cand} is feasible. Then p_{infeas} is replaced by p_{cand}. By doing so for all infeasible p_{infeas}, the whole population becomes feasible, and thus, becomes repaired.

4 Memory Design

Designing memory for improving the quality of solving unconstrained dynamic optimization is an established practice. The usual practice is to set aside a memory space to hold a certain number of individuals (taken from the population). At certain time steps (which are pre–defined or event–triggered) individuals that have performed exceptionally well in solving the dynamic optimization problem are identified and put to the memory, possibly replacing some other using an updating strategy. Later on, if either a change in the fitness landscape is detected [11] and/or the algorithm's performance deteriorates, the stored individuals are retrieved from the memory and inserted into the population, possibly substituting other (poor performing) individuals. By fine–tuning the strategies for identifying suitable individuals, replacing, retrieving, and having the appropriate size of the memory, this approach has been successfully used in solving (unconstrained) dynamic optimization problems, particularly those that show a certain degree of recurrence in its dynamics. However, it is not clear if such an approach is also

a promising option for constrained dynamic optimization problems. As not only the fitness function but also the constraints may change with time, individuals that have been put to the memory because they were good and feasible might, if retrieved, be infeasible. To address this issue, an alternative and new memory strategy for constrained dynamic optimization problems is proposed. It uses some basic ideas from an abstract memory scheme recently introduced [9,10]. Abstract memory is different from conventional memory schemes in two respects. One is that the memory itself is not an extension of the population and hence neither a storing place for individuals (and is therefore likely to have a representation that is different from the individuals'), but it accounts for the spatio–temporal distribution of regions in the search space that are most promising to contain good solutions. Another is the storing and retrieving process.

We briefly recall the memory design, referring to [10] for further details. For setting up the abstract memory, we assume that the bounded search space M has a lower and upper limit $[\ x_{i\,min}\ ,\ x_{i\,max}\ ]$, $i = 1, 2, \ldots, n$ in each direction. We define a grid size ϵ to partition M into rectangular (hyper–) cells. With $h_i = \lceil \frac{x_{i\,max} - x_{i\,min}}{\epsilon} \rceil$ we obtain $\Pi_{i=1}^{n} h_i$ such cells. All cells together are represented by the memory matrix $\mathcal{M}(t) \in \mathbb{R}^{h_1 \times h_2 \times \ldots \times h_n}$, while each cell is specified by the memory matrix element $m_{\ell_1 \ell_2 \ldots \ell_n}(t)$ with $\ell_j = 1, 2, \ldots, h_i$. Each memory matrix element $m_{\ell_1 \ell_2 \ldots \ell_n}(t)$ consists of a counter $count_{\ell_1 \ell_2 \ldots \ell_n}(t)$, which is empty initially, i.e., $count_{\ell_1 \ell_2 \ldots \ell_n}(0) = 0$ for all ℓ_j. For each individual $p_\kappa(t) \in P(t)$ selected to be part of the memorizing, the counter of the element representing the partition cell that the individual belongs to is increased by one. That is, we calculate the index $\ell_i = \lceil \frac{p_{i\,\kappa} - x_{i\,min}}{\epsilon} \rceil$ for all $p_\kappa = (p_{1\kappa}, p_{2\kappa}, \ldots, p_{n\kappa})^T$ and all $1 \leq i \leq n$ and increment the corresponding $count_{\ell_1 \ell_2 \ldots \ell_n}(t)$. Note that this process might be carried out several times in a generation t if more than one individual selected belongs to the same partition. For retrieval we first form an adjunctive memory matrix $\mathcal{M}_\mu(t)$ by dividing $\mathcal{M}(t)$ by the sum of all elements in $\mathcal{M}(t)$, i.e., $\mathcal{M}_\mu(t) = \frac{1}{\sum_{h_i} \mathcal{M}(t)} \mathcal{M}(t)$. Hence, each element in $\mathcal{M}_\mu(t)$ is an approximation of the natural measure $\mu \in [0, 1]$, which expresses the probability that a good solution can be found in the corresponding partition cell $M_{\ell_1 \ell_2 \ldots \ell_n}$ of the search space. We then fix the number of individuals to be generated by τ $(1 \leq \tau \leq \lambda)$ and produce these individuals randomly such that their statistical distribution regarding the partition cells matches the one stored in the memory $\mathcal{M}_\mu(t)$.

The given storage process builds up a probabilistic model of the occurrence of good solutions in the search space. This model is used in the retrieval process to generate randomly new individuals that are likely to drive the evolutionary process towards such promising areas. In the memory design for constrained dynamic optimization problems, we extend this idea of "spatio–temporal cartographing" to the dynamic constraints. Therefore, we put all individuals that have not violated the constraints to a constraint memory matrix $\mathcal{C}(t) \in \mathbb{R}^{h_1 \times h_2 \times \ldots \times h_n}$ using the same process as described above, which results in an adjunctive constraint memory matrix $\mathcal{C}_\mu(t) = \frac{1}{\sum_{h_i} \mathcal{C}(t)} \mathcal{C}(t)$ having equivalent probabilistic properties; see Fig. 1b for a typical image of matrix $\mathcal{C}_\mu(t)$, where the degree of darkness indicates the spatio–temporal probability of a feasible region.

With $\mathcal{M}_\mu(t)$ and $\mathcal{C}_\mu(t)$ we have two memories that represent (an approximation of) the spatio–temporal distribution of (a) likely good candidates for solving the (unconstrained) dynamic optimization problem and (b) likely feasible regions. For using these memories, we consider two schemes, which we will call blending and censoring. For blending, we again fix the number of individuals to be inserted by τ and produce τ_M individuals based on $\mathcal{M}_\mu(t)$ and τ_C individuals based on $\mathcal{C}_\mu(t)$ by the abstract retrieval process given earlier (in the experiments we use $\tau_M = \tau_C$). The $\tau_M + \tau_C = \tau$ individuals are blended together by including all of them into the population. The τ_M retrieved individuals are expected to be nearby likely solutions of the (unconstrained) optimization problem, the τ_C are situated in likely feasible regions. As both groups of individuals together with the remaining population subsequently undergo the constraint processing (either penalty function or repair algorithm) before they are subjected to the evolutionary process of the EA, both aspects of problem hardness (dynamic fitness function and/or dynamic constraints) are addressed separately by blending. So, blending applies the principles that both memories contribute equally and independently to the production of retrieved individuals and that all produced new individuals are blended into the population.

Censoring pursuits a different strategy. Here, we first produce a number of candidate individuals based on the memory matrix $\mathcal{M}_\mu(t)$ alone. This number exceeds the number τ of individuals to be inserted (in the experiments 5τ is used). These candidate individuals are censored by the constraint memory $\mathcal{C}_\mu(t)$ as follows. For each candidate individual it is checked if its corresponding partition cell and its surrounding cells have non–zero entries in $\mathcal{C}_\mu(t)$. All non–zero entries are summed up. The τ individuals with the largest summed-up entries from $\mathcal{C}_\mu(t)$ are inserted into the population. With such a calculation, it is ensured that only those individuals are considered that not only represent likely solutions to the dynamic optimization problem, but also have a high probability of being feasible. Hence, censoring uses both memories hierarchically. The distribution of the individuals regarding the partition cell is fixed entirely based on $\mathcal{M}_\mu(t)$, while $\mathcal{C}_\mu(t)$ is used to fine–tune this selection.

5 Numerical Experiments

In the following, we report three groups of experiments, involving (a) dynamic constraints and a dynamic fitness function, (b) static constraints and a dynamic fitness function, and (c) dynamic constraints and a static fitness function. The dynamics here is a random process with each $c_i(k)$ and/or $cc_j(k)$ for each k being an independent realization of a normally distributed random variable with fixed mean and variance, which makes it recurring in an ergodic sense. We consider the dynamic fitness function (1) with dimension $n = 2$ and the number of cones $N = 7$. The upper and lower bounds of the search space are set to $x_{1\,min} = x_{2\,min} = -3$ and $x_{1\,max} = x_{2\,max} = 3$. The dynamic norm–based constraints (2) use $m = 4$ constraints, see Fig. 1a for illustration. To build the memory $\mathcal{M}_\mu(t)$, the best three individuals of the population take part in the memorizing process for all three generations before a change; all individuals not

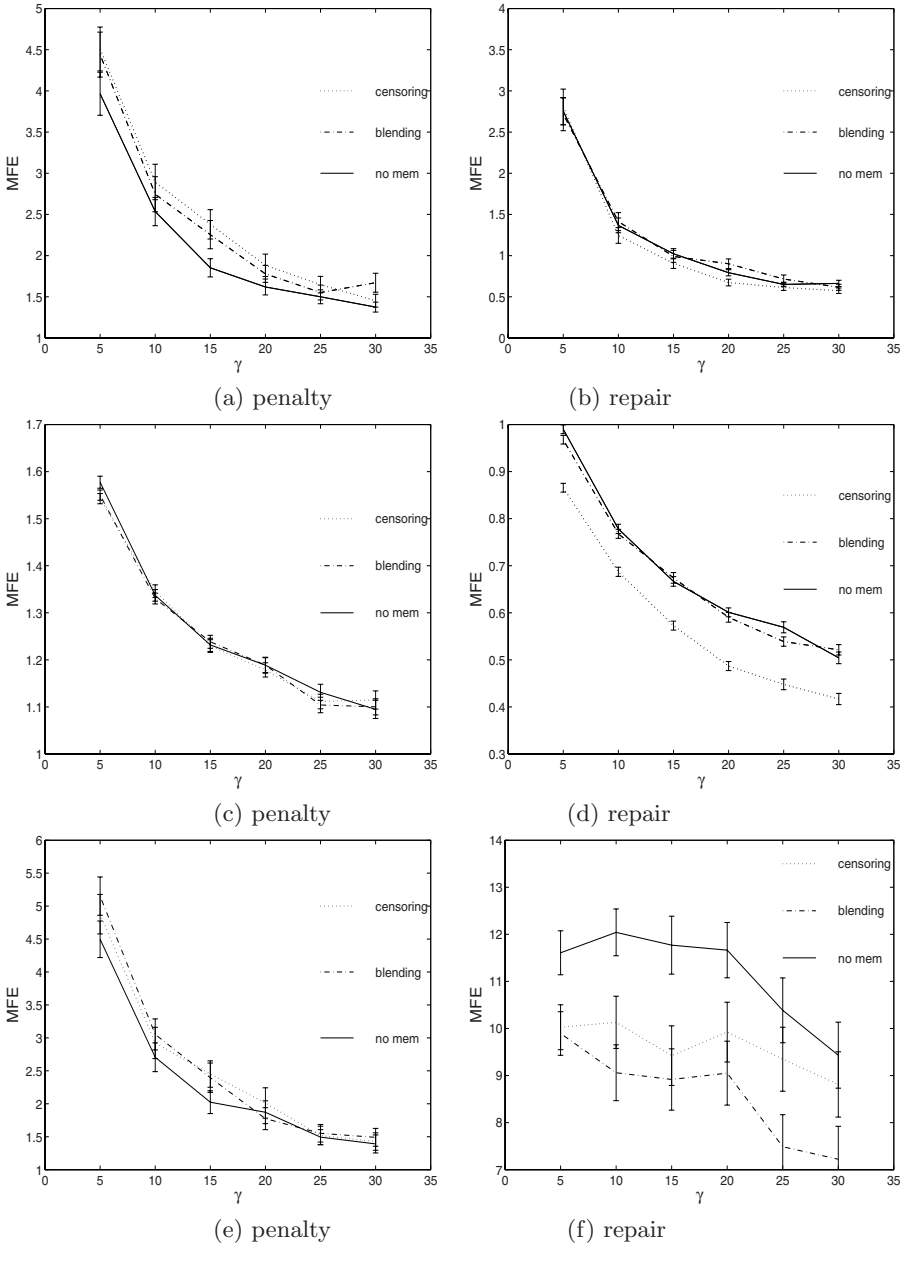


Fig. 2. Memory performance measured by the MFE over change frequency γ: (a,b) dynamic landscape and dynamic constraints, (c,d) dynamic landscape and static constraints (e,f) static landscape and dynamic constraints

violating the constraints participate in composing $\mathcal{C}_\mu(t)$. The employed EA has $\lambda = 40$ individuals, tournament selection of tournament size 2, a fitness–related intermediate sexual recombination (in which λ times two individuals are chosen randomly to produce an offspring that is the fitness–weighted arithmetic mean of both parents) and a standard mutation with the mutation rate 0.1 and hyper–mutation with hyper–mutation rate 5. The memory grid size is $\epsilon = 0.1$; $\tau = 6$ individuals are retrieved and inserted.

Fig. 2 shows the results as the mean fitness error (MFE) for 100 runs and 1000 generations, including the 95% confidence intervals. Penalty functions and repair algorithms are compared for no memory usage and for the proposed memory with both blending and censoring. The results show a better performance of censoring with a repair algorithm in contrast to blending and no memory, but mainly to penalty functions, for dynamic fitness functions and static constraints, and to some extend also for dynamic fitness functions and dynamic constraints. For static landscapes with dynamic constraints the results are generally much poorer, with penalty functions outperforming repair algorithms. A possible explanation for this might be that with static landscapes and dynamic constraints, repair (and hence making all individuals feasible) deprives the evolutionary drive of its flexibility; experimentally observed low diversity and low variety in the percentage of feasible individuals seems to support this view. The experiments have further shown that these results are relatively robust against varying landscape and constraint parameters.

6 Conclusions and Future Work

We proposed a memory design usable to solve constrained dynamic optimization problems. The design was implemented and tested with two schemes, blending and censoring. The results show that for problems with dynamic fitness functions and static constraints, and to some extend also for problems with dynamic fitness functions and dynamic constraints, an improvement is obtained in connection with repair algorithms. The numerical experiments considered only dynamics that is random. Further studies should also treat other kinds of dynamics, for instance regular or chaotic. Besides, the conjecture could be studied that the memory is notwithstanding only sensible if the dynamics of fitness functions and/or constraints shows recurrence in an ergodic sense. For sake of brevity, a detailed analysis of the design parameters of the proposed memory scheme is not reported here, but this should still be done. Finally, setting up a memory inevitably results in initiating learning processes, at least implicitly. So, analysing the learning in the proposed scheme would be another interesting topic for further research.

References

1. Arnold, D.V., Beyer, H.G.: Optimum tracking with evolution strategies. Evol. Comput. 14, 291–308 (2006)
2. Bosman, P.A.N.: Learning and anticipation in online dynamic optimization. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments, pp. 129–152. Springer, Berlin (2007)

3. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Angeline, P.J., et al. (eds.) Proc. Congress on Evolutionary Computation, IEEE CEC 1999, pp. 1875–1882. IEEE Press, Piscataway (1999)
4. Coello Coello, C.A.: Theoretical and numerical constraint–handling techniques used with evolutionary algorithms: a survey of the state of the art. Comp. Meth. Appl. Mech. Eng. 191, 1245–1287 (2002)
5. Mezura–Montes, E.: Constraint–Handling in Evolutionary Optimization. Springer, Berlin (2009)
6. Michalewicz, Z., Nazhiyath, G.: Genocop III: A Co–evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints. In: Michalewicz, Z. (ed.) Proc. 2nd IEEE International Conference on Evolutionary Computation, vol. 2, pp. 647–651 (1995)
7. Morrison, R.W., De Jong, K.A.: Triggered hypermutation revisited. In: Zalzala, A., et al. (eds.) Proc. Congress on Evolutionary Computation, IEEE CEC 2000, pp. 1025–1032. IEEE Press, Piscataway (2000)
8. Nguyen, T.T., Yao, X.: Benchmarking and solving dynamic constrained problems. In: Tyrrell, A. (ed.) Proc. Congress on Evolutionary Computation, IEEE CEC 2009, pp. 690–697. IEEE Press, Piscataway (2009)
9. Richter, H., Yang, S.: Memory based on abstraction for dynamic fitness functions. 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., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 596–605. Springer, Heidelberg (2008)
10. Richter, H., Yang, S.: Learning behavior in abstract memory schemes for dynamic optimization problems. Soft Computing 13, 1163–1173 (2009)
11. Richter, H.: Detecting change in dynamic fitness landscapes. In: Tyrrell, A. (ed.) Proc. Congress on Evolutionary Computation, IEEE CEC 2009, pp. 1613–1620. IEEE Press, Piscataway (2009)
12. Salcedo–Sanz, S.: A survey of repair methods used as constraint handling techniques in evolutionary algorithms. Comp. Sci. Rev. 3, 175–192 (2009)
13. Simões, A., Costa, E.: Variable-size memory evolutionary algorithm to deal with dynamic environments. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 617–626. Springer, Heidelberg (2007)
14. Simões, A., Costa, E.: Evolutionary algorithms for dynamic environments: Prediction using linear regression and Markov chains. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 306–315. Springer, Heidelberg (2008)
15. Singh, H.K., Isaacs, A., Nguyen, T.T., Ray, T., Yao, X.: Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems. In: Tyrrell, A. (ed.) Proc. Congress on Evolutionary Computation, IEEE CEC 2009, pp. 3127–3134. IEEE Press, Piscataway (2009)
16. Tinós, R., Yang, S.: A self–organizing random immigrants genetic algorithm for dynamic optimization problems. Genet. Program. Evol. Mach. 8, 255–286 (2007)
17. Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 788–799. Springer, Heidelberg (2006)

Multi-population Genetic Algorithms with Immigrants Scheme for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks

Hui Cheng and Shengxiang Yang

Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
{hc118,s.yang}@mcs.le.ac.uk

Abstract. The static shortest path (SP) problem has been well addressed using intelligent optimization techniques, e.g., artificial neural networks, genetic algorithms (GAs), particle swarm optimization, etc. However, with the advancement in wireless communications, more and more mobile wireless networks appear, e.g., mobile ad hoc network (MANET), wireless mesh network, etc. One of the most important characteristics in mobile wireless networks is the topology dynamics, that is, the network topology changes over time due to energy conservation or node mobility. Therefore, the SP problem turns out to be a dynamic optimization problem in mobile wireless networks. In this paper, we propose to use multi-population GAs with immigrants scheme to solve the dynamic SP problem in MANETs which is the representative of new generation wireless networks. The experimental results show that the proposed GAs can quickly adapt to the environmental changes (i.e., the network topology change) and produce good solutions after each change.

1 Introduction

A mobile ad hoc network (MANET) [11] is a self-organizing and self-configuring multi-hop wireless network, comprised of a set of mobile hosts (MHs) that can move around freely and cooperate in relaying packets on behalf of each other. In this paper, we investigate the shortest path (SP) routing, which concerns with finding the shortest path from a specific source to a specific destination in a given network while minimizing the total cost associated with the path. The SP problem has been investigated extensively. It involves a classical combinatorial optimization problem arising in many design and planning contexts [1,2].

There are several search algorithms for the SP problem: the Dijkstra's algorithm, the breadth-first search algorithm and the Bellman-Ford algorithm, etc. All these algorithms have polynomial time complexity. Therefore, they will be effective in fixed infrastructure wireless or wired networks. But, they exhibit unacceptably high computational complexity for real-time communications involving rapidly changing network topologies [2,3]. Since the algorithms with polynomial time complexity are not suitable for the real-time computation of shortest paths, quite a few research work have been conducted to solve SP problems using artificial intelligence techniques, e.g., artificial neural networks (ANNs) [2], genetic algorithms (GAs) [3], and particle swarm optimization (PSO) [7].

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 562–571, 2010.

However, so far all these algorithms mainly address the static SP problem. When the network topology changes, they will regard it as a new network and restart the solving process over the new topology. As is well known that the topology changes rapidly in MANETs due to the characteristics of wireless networks, e.g., battery exhaustion and node mobility. Therefore, for the dynamic SP problem in MANETs, these algorithms are not good choices since they require frequent restart and cannot meet the real-time requirement. Therefore, for the dynamic SP problem in a changing network environment, we need to employ new appropriate approaches.

In recent years, studying EAs for DOPs has attracted a growing interest due to its importance in EA's real world applications [14]. The simplest way of addressing DOPs is to restart EAs from scratch whenever an environment change is detected. Although the restart scheme really works for some cases [13], for many DOPs it is more efficient to develop other approaches that make use of knowledge gathered from old environments. One of the possible approaches is to maintain and reintroduce diversity during the run of EAs, i.e., the immigrants schemes [15]. Multi-population approach [4] is also an effective technique for DOPs. In the multi-population GA (MPGA), some populations are responsible for exploiting and others for exploring. By both exploiting and exploring the solution space, MPGA can well adapt to the environmental changes.

In this paper, the multi-population GA with immigrants scheme is implemented and applied to solve the dynamic SP problem. The algorithm is denoted as iMPGA. A large population is created, which will split into several small populations after evolving for a certain time. These small populations continue the search by either exploiting or exploring the solution space. Once the topology is changed, all the small populations are processed in an appropriate way and then merge together. At each generation, to enhance the diversity a small number of random immigrants are added into the single population or the small populations which are responsible for exploring. This process is repeated for each change interval. Since end-to-end delay [10] is a pretty important quality-of-service (QoS) metric to guarantee the real-time data delivery, we also require the routing path to satisfy the delay constraint. For comparison purposes, we also implement the Standard GA (SGA), the Restart GA, and the random immigrants GA (RIGA). By simulation experiments, we evaluate their performance on the dynamic SP problem. The results show that iMPGA significantly outperforms the other three GA methods.

2 Model

In this section, we first present our network model and then formulate the problem of dynamic SP routing. We consider a MONET operating within a fixed geographical region. We model it by a undirected and connected topology graph $G_0(V_0, E_0)$, where V_0 represents the set of wireless nodes (i.e., routers) and E_0 represents the set of communication links connecting two neighboring routers falling into the radio transmission range. A communication link (i, j) can not be used for packet transmission until both node i and node j have a radio interface each with a common channel. However, the channel assignment is beyond the scope of this paper. In addition, message transmission on a wireless communication link will incur remarkable delay and cost.

Here, we summarize some notations that we use throughout this paper.

– $G_0(V_0, E_0)$, the initial MANET topology graph.

- $G_i(V_i, E_i)$, the MANET topology graph after the ith change.
- s, the source node.
- r, the destination node.
- $P_i(s, r)$, a path from s to r on the graph G_i.
- d_l, the transmission delay on the communication link l.
- c_l, the cost on the communication link l.
- $\Delta(P_i)$, the total transmission delay on the path P_i.
- $C(P_i)$, the total cost of the path P_i.

The problem of the dynamic SP routing can be informally described as follows. Initially, given a network of wireless routers, a delay upper bound, a source node and a destination node, we wish to find a delay-bounded least cost loop-free path on the topology graph. Then periodically or stochastically, due to energy conservation or some other issues, some nodes are scheduled to sleep or some sleeping nodes are scheduled to wake up. Therefore, the network topology changes from time to time. The objective of our problem is to quickly find the new optimal delay-constrained least cost acyclic path after each topology change.

More formally, consider a mobile ad hoc network $G(V, E)$ and a unicast communication request from the source node s to the destination node r with the delay upper bound Δ. The *dynamic delay-constrained shortest path problem* is to find a series of paths $\{P_i | i \in \{0, 1, ...\}\}$ over a series of graphs $\{G_i | i \in \{0, 1, ...\}\}$, which satisfy the delay constraint as shown in (1) and have the least path cost as shown in (2).

$$\Delta(P_i) = \sum_{l \in P_i(s,r)} d_l \leq \Delta \,. \tag{1}$$

$$C(P_i) = \min_{P \in G_i} \left\{ \sum_{l \in P(s,r)} c_l \right\} \,. \tag{2}$$

3 Design of GA for SP Problem

This section describes the design of the GA for the SP problem. The GA operations consist of several key components: genetic representation, population initialization, fitness function, selection scheme, crossover and mutation. A routing path consists of a sequence of adjacent nodes in the network. Hence, it is a natural choice to adopt the path-oriented encoding method. For the routing problems, the path-oriented encoding and the path-based crossover and mutation are also very popular [3]. For the selection scheme, the pair-wise tournament selection without replacement [6] is employed and the tournament size is 2.

3.1 Genetic Representation

A routing path is encoded by a string of positive integers that represent the IDs of nodes through which the path passes. Each locus of the string represents an order of a node (indicated by the gene of the locus). The gene of the first locus is for the source node and the one of the last locus is for the destination node. The length of a routing path should not exceed the maximum length $|V_0|$, where V_0 is the set of nodes in the MANET. Chromosomes are encoded under the delay constraint. In case it is violated, the encoding process is usually repeated so as to satisfy the delay constraint.

3.2 Population Initialization

In GA, each chromosome corresponds to a potential solution. The initial population Q is composed of a certain number, denoted as n, of chromosomes. To explore the genetic diversity, in our algorithm, for each chromosome, the corresponding routing path is randomly generated. We start to search a random path from s to r by randomly selecting a node v_1 from $N(s)$, the neighborhood of s. Then we randomly select a node v_2 from $N(v_1)$. This process is repeated until r is reached. Thus, we get a random path $P(s, r)=\{s, v_1, v_2, ..., r\}$. Since the path should be loop-free, the nodes that are already included in the current path are excluded, thereby avoiding reentry of the same node. The initial population is generated as follows.

Step 1: Start(j=0).

Step 2: Generate chromosome Ch_j: search a random loop-free path $P(s, r)$;

Step 3: j=j+1. If $j < n$, go to *Step 2*, otherwise, stop.

Thus, the initial population $Q = \{Ch_0, Ch_1, ..., Ch_{n-1}\}$ is obtained.

3.3 Fitness Function

Given a solution, we should accurately evaluate its quality (i.e., fitness value), which is determined by the fitness function. In our algorithm, we aim to find the least cost path between the source and the destination. Our primary criterion of solution quality is the path cost. Therefore, among a set of candidate solutions (i.e., unicast paths), we choose the one with the least path cost. The fitness value of chromosome Ch_j (representing the path P), denoted as $F(Ch_j)$, is given by:

$$F(Ch_j) = [\sum_{l \in P(s,r)} c_l]^{-1} . \tag{3}$$

The proposed fitness function only involves the total path cost. As mentioned above, The delay constraint is checked for each chromosome in the course of the run.

3.4 Crossover and Mutation

GA relies on two basic genetic operators - crossover and mutation. Crossover processes the current solutions so as to find better ones. Mutation helps GA keep away from local optima [3]. The performance of GA depends on them greatly. The type and implementation of operators depend on problem-specific encoding.

In our algorithm, since chromosomes are expressed by the path structure, we adopt single point crossover to exchange partial chromosomes (subpath) at positionally independent crossing sites between two chromosomes [3]. With the crossover probability, each time we select two chromosomes Ch_i and Ch_j for crossover. Ch_i and Ch_j should possess at least one common node. Among all the common nodes, one node, denoted as v, is randomly selected. In Ch_i, there is a path consisting of two parts: ($s \xrightarrow{Ch_i} v$) and ($v \xrightarrow{Ch_i} r$). In Ch_j, there is a path consisting of two parts: ($s \xrightarrow{Ch_j} v$) and ($v \xrightarrow{Ch_j} r$). The crossover operation exchanges the subpaths ($v \xrightarrow{Ch_i} r$) and ($v \xrightarrow{Ch_j} r$).

The population will undergo the mutation operation after the crossover operation is performed. With the mutation probability, each time we select one chromosome Ch_i on

which one gene is randomly selected as the mutation point (i.e., mutation node), denoted as v. The mutation will replace the subpath ($v \xrightarrow{Ch_i} r$) by a new random subpath.

Both crossover and mutation may produce new chromosomes which are infeasible solutions. Therefore, we check if the paths represented by the new chromosomes are acyclic. If not, repair functions [8] will be applied to eliminate the loops. Here the detail is omitted due to the space limit. All the new chromosomes produced by crossover or mutation satisfy the delay constraint since it has already been considered.

4 iMPGA: Multi-population GAs with Immigrants Scheme

The random immigrants approach was proposed by Grefenstette [5] with the inspiration from the flux of immigrants that wander in and out of a population between two generations in nature. It maintains the population diversity level by replacing some individuals of the current population with random individuals, called random immigrants, every generation. As to which individuals in the population should be replaced, usually there are two strategies: replacing random individuals or replacing the worst ones. In order to avoid that random immigrants disrupt the ongoing search progress too much, especially during the static period between two environmental changes, the ratio of the number of random immigrants to the population size, r_i, is usually set to a small value.

The traditional genetic algorithm has a single population searching through the entire search space. Multi-population approach tries to divide the search space into several parts and then uses a number of small populations to search them separately. Normally, one of the small populations acts as the parent population or the core population. In the Forking genetic algorithms (FGAs) [12], the parent population continuously searches for new optimum, while a number of child populations try to exploit previously detected promising areas. In the Shifting Balance GA [9], the core population is used to exploit the best solution found, while the colony populations are responsible for exploring different areas in the solution space.

In this paper, we generally follow the idea of the FGAs. However, to address the dynamic SP problem, we still need to make specific design in our algorithm. To measure the similarity degree between two individuals, we define the distance between any two individuals by counting the same links shared by them. The more same links they share, the closer they are. For the parent population which is responsible for exploring, we expect that the individuals in it are kept far away from each other in the distance. Thus, the population can search a wide area. For a child population which is responsible for exploiting, we expect that the individuals in it stay close to an optimum and perform lots of local search.

In iMPGA, initially we randomly generate a large single population. For each given change interval I, the whole population will evolve together for $\lfloor I/2 \rfloor$ generations. Then the single population is split into three small populations. Of them, one small population will act as the parent population for exploring and the other two will act as the child populations for exploiting. To achieve this goal, we develop the following splitting method. First, we identify the present optimal individual $Pop1_{opt}$ in the whole population. Then we find its closest neighbor $Pop1_1$, 2nd closest neighbor $Pop1_2$, 3rd closest neighbor $Pop1_3$, till the (m-1)th closest neighbor $Pop1_{m-1}$. All these m individuals form the first

child population $\{Pop1_{opt}, Pop1_1, Pop1_2, Pop1_3, ..., Pop1_{m-1}\}$. Among all the remaining individuals of the whole population, the optimal one is identified again, denoted as $Pop2_{opt}$. Similarly, among the remaining individuals, we determine its closest neighbor $Pop2_1$, 2nd closest neighbor $Pop2_2$, 3rd closest neighbor $Pop2_3$, till the (m-1)th closest neighbor $Pop2_{m-1}$. All these m individuals form the second child population $\{Pop2_{opt}, Pop2_1, Pop2_2, Pop2_3, ..., Pop2_{m-1}\}$. All the remaining individuals form the third population, i.e., the parent population.

These three small populations keep evolving independently till the change interval ends. When a new change is detected, i.e., the topology is modified, all of them need to be processed appropriately and then merged together in order to form a single population again. We develop the following processing method for these small populations to adapt to the environmental changes. For each of them, if the optimal individual in it becomes infeasible, the whole population will be replaced by random immigrants. Otherwise, if the optimal individual in it is feasible, only the infeasible individuals in the population will be replaced by random immigrants. The reason to do so is that if the optimal individual becomes infeasible, all the other individuals are also infeasible and therefore the whole population should be abandoned. However, if the optimal individual is suitable for the new environment, we also want to keep other individuals which are also suitable for the new environment. Thus, the useful information in the old environment can be reused to guide the search in the new environment.

In our algorithm, at each generation, a small number of random immigrants are added into the population. Before the splitting of the population, all the random immigrants are imported into the single population to replace the worst ones. After the splitting, all the random immigrants are only imported into the parent population since it is responsible for exploring.

5 Experimental Study

We implement iMPGA, RIGA, SGA, and Restart GA for the dynamic SP problem by simulation. For RIGA and SGA, if the change makes one individual in the current population become infeasible (e.g., one link in the corresponding path is lost after the change), we add penalty value to that individual. By simulation experiments, we evaluate their performance in a continuously changing mobile ad hoc network.

5.1 Experimental Design

All the algorithms start from the initial network topology of 100 nodes. Then every I generations, the present best path is identified and a certain number (say, U) of links on the path are selected for removal. It means that the selected links will be forced to be removed from the network topology. However, just before the next change occurs, the network topology will be recovered to its original state and ready for the oncoming change. The population is severely affected by each topology change since the optimal solution and possibly some other good solutions become infeasible suddenly. Considering that the optimal path length could not be a large number, we let U range from 1 to 3 to see the effect of the change severity. Under this network dynamics model, the

topology series cannot be generated in advance because every change is correlated with the running of the algorithm. We allow 10 changes in each run of the algorithms. We set up experiments to evaluate the impact of the change interval and the change severity, and the improvements over traditional GAs and RIGA.

In all the experiments, the whole population size n is set to 100, the child population size m is set to 20, and the mutation probability is set to 0.1. For the random immigrants scheme, r_i is set to 0.2. In addition, we set the number of changes to 19 and therefore the algorithms will work over 20 different but highly-correlated network topologies (the initial topology plus the 19 changed topologies). Both the source and destination nodes are randomly selected and they are not allowed to be scheduled in any change. The delay upper bound Δ is set to be 2 times of the minimum end-to-end delay.

5.2 Experimental Results and Analysis

At each generation, for each algorithm, we select the best individual from the current population and output the cost of the shortest path represented by it. We repeat each experiment 10 times and get the average values of the best solutions at each generation. First, we investigate the impact of the change interval on the algorithm performance. We set I to 5, 10, and 15 separately to see the impact of change interval (i.e., change frequency) on the algorithm performance. Here the number of links removed per change is fixed to 2.

When the change interval is 5, the population evolves only 5 generations between two sequential changes. Intuitively, a larger interval will give the population more time to evolve and search better solutions than what a smaller interval does. We compare the quality of solutions obtained by iMPGA at different intervals. However, one problem is that the total generations are different for different intervals, i.e., 100, 200 and 300 versus the interval 5, 10, and 15 when there are 20 different topologies. Since the number of change points (i.e., the generation at which a new topology is applied) is the same for all the intervals, we take the data at each change point and its left two and right two generations. Thus, the three different data sets can be aligned over the three different intervals. Fig. 1 shows the comparison results in terms of the change intervals.

Since the generation number does not correspond to the actual number when the interval is 10 or 15, we rename it as pseudo generation. From the two subfigures, it can be seen that the solution quality becomes better when the change interval is increased from 5 to 10. However, when the change interval is increased from 10 to 15, the results in both subfigures are slightly different. In Fig. 1(a), the iMPGA shows competing performance for both intervals. For five times, the performance at interval 10 is better and for the other five times, the performance at interval 15 is better. In Fig. 1(b), the performance at interval 15 is better than the performance at interval 10 for all the times. The reason is that in Fig. 1(b), the generations that the whole population has evolved at interval 15 are much larger than the generations that the whole population has evolved at interval 10. Longer evolution brings better solutions. Therefore, the capability of the multi-population genetic algorithm in searching the optimum has been significantly enhanced. In traditional GA, the population may converge after evolving for a while. However, in iMPGA, due to the introduction of random immigrants, the population can keep evolving and get out of the trap in the local optimum.

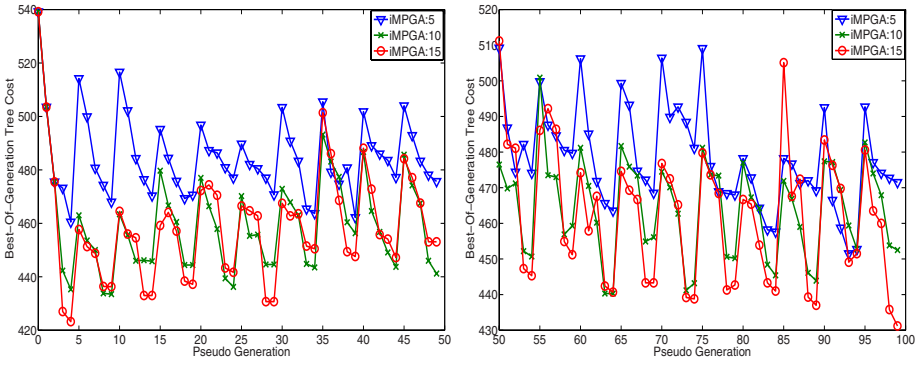


Fig. 1. Comparison of the solution quality of iMPGA with different change intervals from (a) generation 0-49 and (b) generation 50-99

To evaluate the effect of the change severity on the algorithm performance, we vary the number of links removed per change from 1 to 3. Meanwhile, the change interval is fixed to 10 since it is a reasonably good change frequency as shown in the above experiments. With more links removed from the network, the environmental changes become more severe. Furthermore, since all the removed links come from the present best path, some individuals including the optimal one in the population become infeasible. It is also possible that the whole population becomes infeasible if each individual contains at least one removed link. The more the links removed, the higher the probability of an individual being infeasible.

Fig. 2 shows the comparison results in terms of the change severities. It can be seen that the quality of solution is the best when the number of links removed per change is 1 and the worst when the number is 3. However, the difference between iMPGA:1 and iMPGA:2 is less significant than the difference between iMPGA:2 and iMPGA:3. The reason is that the increase in the number of links removed per change is not proportional to the increase in the change severity. To remove one more link will bring a much higher change severity to the network and therefore affect much more individuals in the population. Another interesting point is that in Fig. 2(b), the performance differences between the algorithms with different change severities become less than the differences in Fig. 2(a). It is also due to the enhanced search capability of the multi-population algorithm after long time evolution as explained above.

The quality of solution is the most important metric to evaluate the algorithm performance. We compare iMPGA with both traditional GAs and random immigrants GA. The two traditional GAs are Standard GA and Restart GA. We set the change interval to 10 and the number of links removed per change to 2, respectively. Since iMPGA is a dynamic GA which is specifically designed for the dynamic environment, it should show better performance than the traditional GAs over our dynamic shortest-path problem. Fig. 3(a) shows the comparison results between iMPGA and traditional GAs. It can be seen that iMPGA achieves better solutions than both of the traditional GAs. Restart GA shows the worst performance due to frequent restart which does not give the

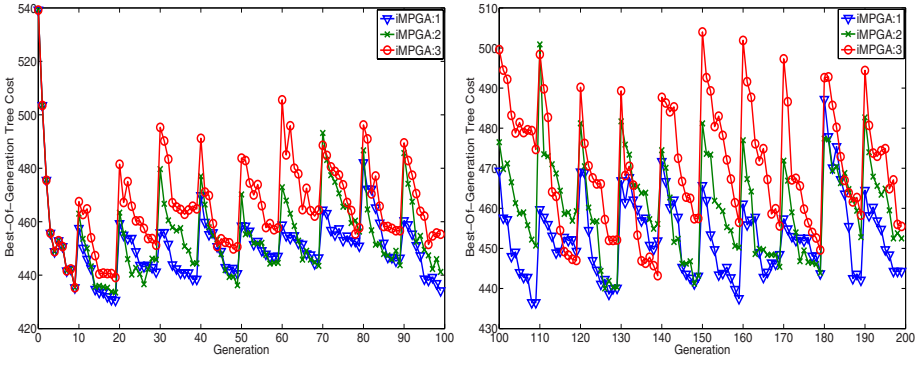


Fig. 2. Comparison of the solution quality of iMPGA with different change severities from (a) generation 0-99 and (b) generation 100-199

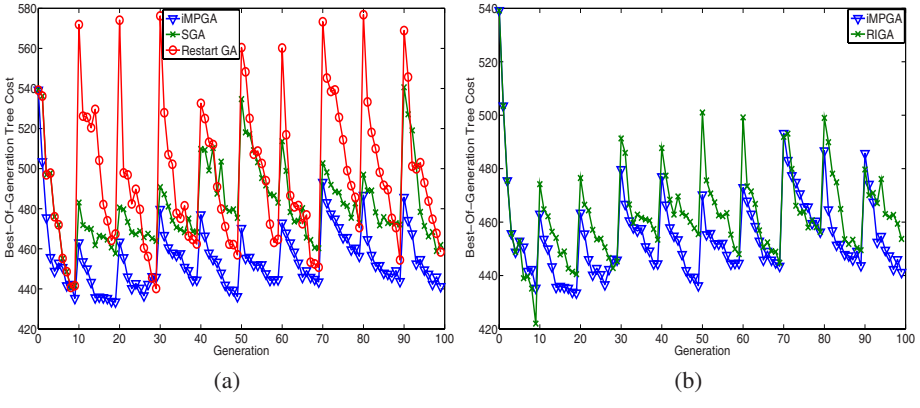


Fig. 3. Comparison of the solution quality of iMPGA against (a) traditional GAs and (b) RIGA

population enough time to evolve. Although RIGA is also a dynamic GA, it does not utilize the approach of multiple populations to help search. Fig. 3(b) shows the comparison results between iMPGA and RIGA. It shows that iMPGA performs better than RIGA. This verifies that the multi-population approach helps improve the capability of GA in handling dynamic environment.

6 Conclusions

The static SP problem considers the static network topology only. Intuitively, it is a much more challenging task to deal with the dynamic SP problem in a rapidly changing network environment such as MANETs than to solve the static one in a fixed infrastructure. Recently, there has been a growing interest in studying GAs for dynamic optimization problems. Among approaches developed for GAs to deal with DOPs, the

multi-population GA aims at handling the problem dynamics by using multiple small populations to perform both exploration and exploitation. Random immigrants scheme is another approach which maintains the diversity of the population throughout the run via introducing new individuals into the current population. In this paper, we propose iMPGA which combines both the multi-population approach and immigrants. We well design the GA components for the SP problem and the multi-population GA with immigrants scheme. Simulation experiments are conducted in a large scale MANET. The results show that iMPGA is a powerful technique for solving the dynamic SP problem and has potential to be applied to the real-world telecommunication network. In the future work, we will further investigate the robustness of the solutions provided by us.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1.

References

1. Ali, M.K., Kamoun, F.: Neural networks for shortest path computation and routing in computer networks. IEEE Trans. on Neural Networks 4(6), 941–954 (1993)
2. Ahn, C.W., Ramakrishna, R.S., Kang, C.G., Choi, I.C.: Shortest path routing algorithm using hopfield neural network. Electronics Letters 37(19), 1176–1178 (2001)
3. Ahn, C.W., Ramakrishna, R.S.: A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans. on Evol. Comput. 6(6), 566–579 (2002)
4. Branke, J., Kaußler, T., Schmidt, C., Schmeck, H.: A multi-population approach to dynamic optimization problems. In: Proc. 4th Int. Conf. on Adaptive Computing in Design and Manufacture, pp. 299–308 (2000)
5. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137–144 (1992)
6. Lee, S., Soak, S., Kim, K., Park, H., Jeon, M.: Statistical properties analysis of real world tournament selection in genetic algorithms. Applied Intell. 28(2), 195–205 (2008)
7. Mohemmed, A.W., Sahoo, N.C., Geok, T.K.: Solving shortest path problem using particle swarm optimization. Applied Soft Comput. 8(4), 1643–1653 (2008)
8. Oh, S., Ahn, C., Ramakrishna, R.: A genetic-inspired multicast routing optimization algorithm with bandwidth and end-to-end delay constraints. In: Proc. 13th Int. Conf. on Neural Information Processing, pp. 807–816 (2006)
9. Oppacher, F., Wineberg, M.: The shifting balance genetic algorithm: improving the GA in a dynamic environment. In: Proc. Genetic and Evol. Comput. Conf., pp. 504–510 (1999)
10. Parsa, M., Zhu, Q., Garcia-Luna-Aceves, J.: An iterative algorithm for delay-constrained minimum-cost multicasting. IEEE/ACM Trans. on Networking 6(4), 461–474 (1998)
11. Perkins, C.E. (ed.): Ad Hoc Networking. Addison-Wesley, London (2001)
12. Tsutsui, S., Fujimoto, Y., Ghosh, A.: Forking genetic algorithms: GAs with search space division schemes. Evol. Comput. 5(1), 61–80 (1997)
13. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)
14. Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. on Evol. Comput. 12(5), 542–561 (2008)
15. Yang, S.: Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evol. Comput. 16(3), 385–416 (2008)

Measuring Fitness Degradation in Dynamic Optimization Problems

Enrique Alba and Briseida Sarasola

Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, 29071 Málaga, Spain

Abstract. Measuring the performance of algorithms over dynamic optimization problems (DOPs) presents some important differences when compared to static ones. One of the main problems is the loss of solution quality as the optimization process advances in time. The objective in DOPs is tracking the optima as the landscape changes; however it is possible that the algorithm fails to follow the optima after some changes happened. The main goal in this article is to introduce a new way of measuring how algorithms are able to maintain their performance during the dynamic optimization process. We propose a measure based on linear regression and study its behaviour. In order to do so, we propose a scenario based on the moving peaks benchmark and analyze our results using several metrics existing in the literature. We test our measure for degradation on the same scenario, applying it over accuracy values obtained for each period, and obtain results which help us to explain changes in algorithm performances.

Keywords: Performance measures, dynamic optimization, moving peaks.

1 Introduction

The problem of finding good performance measures for dynamic optimization problems (DOPs) is not a trivial one. A good measure should at least describe what the researcher is actually perceiving. It should have also a lower set of restrictions (in oder to be widely applicable) and allow a numerical (maybe statistical) treatment of its results. Some traditional metrics from non-stationary problems, like offline performance [2] and accuracy [9], have been transferred to DOPs, although they often need to be modified and adapted to dynamic environments. Other metrics have been specially designed for DOPs, like collective mean fitness [8]. Although there is a wide plethora of other measures available (window accuracy [9], best known peak error [3], and peak cover [2]), most current studies tend to use the three ones mentioned in the first place, as well as visual analysis of the algorithm running performance.

However, at this moment there is no general consensus about which metric to use. Furthermore, most metrics focus on one aspect of the optimization process, namely the solution quality, while few metrics explore other aspects of the problem dynamics. Among these other metrics we can cite the ones reporting

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 572–581, 2010.

the diversity (most typically entropy and inertia [7]), as well as stability and ε-reactivity [9]. An important issue which is usually not taken into account in existing studies is the ability of a certain algorithm to obtain good solutions for a long time in a maintained manner while the search landscape changes, i.e. to be able to track the moving optima for a big number of periods. A period is an interval of time without changes in the problem definition. Based in existing results and in our own experience, the performance of an algorithm over contiguous and continuous changes in the problem degrades with time. Our objective in this work is then to propose a new metric to measure how an algorithm degrades as the search advances. For that purpose, we consider a scenario using the moving peaks benchmark and genetic algorithms. This is a fundamental issue since it characterizes the ability of the algorithm in the long term, giving the actual power of the algorithm for a real application or for its scalability in time.

The rest of the article is structured as follows: Section 2 explains the main performance measures in the literature; Section 3 exposes the moving peaks problem, which will be used as case study in this paper; Section 4 analyses how existing measures perform for a given case study; Section 5 presents a way of measuring fitness degradation; finally, the conclusions are drawn in Section 6.

2 Existing Performance Metrics for DOPs

The aim in dynamic optimization is not only to find the global optima, but to be able to track the movement of these optima through the search time. Although it is still quite common in existing works to use line charts to visually compare the running fitness of algorithms, a number of numeric measures have been proposed. In this article we discuss the most widely used metrics in the literature.

Offline performance. Its usage for DOPs was proposed in [2]. It is calculated as the average of the best value found so far in the current period (see Eq. 1). It requires that changes in the landscape are known beforehand for its computation.

$$x^* = (1/N) \cdot \sum_{i=1}^{N} f(period\_best_i) \tag{1}$$

Collective mean fitness. It was introduced by Morrison in [8]. It is similar to offline performance, but considers the best value in the current generation, and thus does not require to know about changes in the search space (see Equation 2).

$$F_C = (1/N) \cdot \sum_{i=1}^{N} f(generation\_best_i) \tag{2}$$

Accuracy, stability, and reactivity. This group of three measures were first proposed for static optimization problems, and adapted for dynamic environments in [9]. The accuracy measures how good the best solution in the current population is with respect to the best (Max_F) and worst (Min_F) known values in

the search space. It ranges between 0 and 1, where a value closer to 1 means a higher accuracy level. The formula is often simplified with $Min_F = 0$. This measure is also known as relative error.

$$accuracy_i = \frac{f(generation\_best_i) - Min_F}{Max_F - Min_F} \tag{3}$$

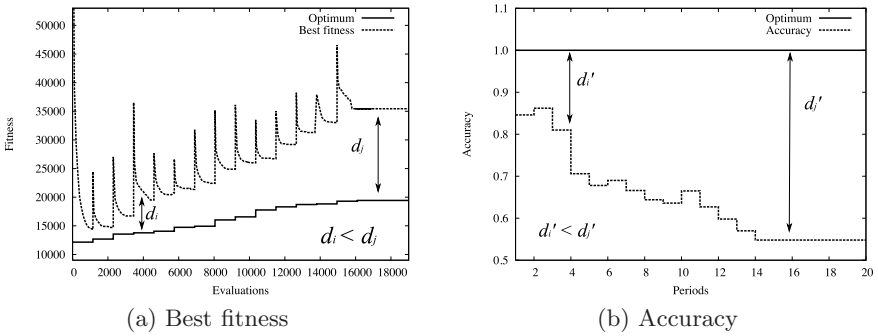
Stability is also viewed as an important issue in DOPs. An algorithm is stable if the changes in the environment do not affect its accuracy severely (see Eq. 4).

$$stability_i = \max\{0, accuracy_i - accuracy_{i-1}\}\,, \tag{4}$$

where $stability \in [0, 1]$. An algorithm is considered stable if stability is close to 0. Finally, another aspect to be taken into account is the ability of the algorithm to react quickly to changes. This is measured by the ε-reactivity, which ranges between 1 and the number of generations ($maxgen$) (a smaller value implies a higher reactivity):

$$reactivity_i = \min\{i' - i | i < i' \leq maxgen, i \in \mathbb{N}, \frac{accuracy_{i'}}{accuracy_i} \geq (1 - \varepsilon)\} \tag{5}$$

However, all these metrics do not reflect a very important aspect in DOPs. Algorithm performance can degrade after the landscape has changed several times, resulting in a loss of fitness quality in the following optimization stages. Fig. 1 shows an example for an algorithm over a general problem. On the left, the running best fitness and the known optimum are represented. It can be easily seen that the running best fitness is much closer to the known best fitness at first periods (d_i), while the distance between them becomes bigger at the last periods (d_j). The same situation is illustrated in the graph on the right, but this time we represent the accuracy for each period compared to the maximum possible accuracy value ($accuracy = 1.0$).



(a) Best fitness (b) Accuracy

Fig. 1. An example to illustrate degradation in DOPs

3 The Moving Peaks Problem

There are several benchmark problems for DO, such as the dynamic bit-matching, royal road, moving parabola, time varying knapsack problem, etc. In this paper, we focus on one of the most widely used benchmark problem for DO: the moving peaks problem [2]. The moving peaks idea is to have an artificial multi-dimensional landscape consisting of several peaks where the height, width, and position of each peak is altered every time a change in the environment occurs. The cost function for N dimensions and m peaks has the following form:

$$F(\bar{\boldsymbol{x}}, t) = \max\{B(\bar{\boldsymbol{x}}), \max_{i=1...m} P(\bar{\boldsymbol{x}}, h_i(t), w_i(t), \bar{\boldsymbol{p}}_i(t))\} \tag{6}$$

where $B(\bar{\boldsymbol{x}})$ is a time-invariant "base" landscape, and P is the function defining a peak shape, where each of the m peaks has its own time-varying parameters height (h), width (w), and location ($\bar{\boldsymbol{p}}$). Every certain number of evaluations, the height and width of every peak are changed by adding a random Gaussian variable. The location of every peak is changed by a vector v of fixed length s. A parameter λ determines if a peak change depends on the previous move or not.

4 Solving the Moving Peaks Problem

This section is aimed at studying the behaviour of the metrics explained in Section 2. For that purpose, we consider the moving peaks problem, using a problem configuration which corresponds closely to the standard settings proposed by Branke[1]. We use a plane defined in $(0, 100) \times (0, 100)$ with 10 dimensions and 10 peaks. The peak heights are defined in the interval [30, 70] and the widths in [0.001, 0.2]. The height change severity is set to 7.0 and the width change severity to 0.01. Changes occur every 1000 evaluations and $\lambda = 0$.

We use an elitist generational genetic algorithm (genGA) as well as three well-known strategies which adapt metaheuristics to dynamic environments: hypermutation (hm) [4], memory (me) [6], and random immigrants (ri) [5]. This results in four algorithms: genGA, genGA+hm, genGA+me, genGA+ri. The population size is 100 individuals, parent selection is done by binary tournament, single point crossover probability p_c equals 1.0, and mutation probability for each solution is $p_m = 0.1$.

We start by comparing the graphs for a mean execution of each algorithm, which is the most basic strategy (Section 4.1). Then we analyze the results with Morrison's collective mean fitness and Weicker's measures (see sections 4.2 and 4.3 respectively). The offline performance results are excluded since they are identical to those of collective mean fitness (due to the elitist genGA).

4.1 Studying the Graphical Representation of Mean Executions

The most immediate way of analyzing the performance of algorithms is comparing the graphs which show the running fitness. This is done in most existing

[1] http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/

works. The advantage is a fast an intuitive way of comparing performances; however, the results can often be difficult to interpret, they might not always fit in a figure of the desired size, and they can easily lead to confusion.

Fig. 2 shows an example of this procedure, where the mean of all 100 runs for each algorithm is represented. The length of the run has been shortened to $40k$ evaluations for the sake of clarity. From this graph we conclude that genGA+ri gives usually better fitness values than the rest; genGA+me and canonical genGA obtain similar results in the first half of the simulation, although genGA+me comes closer to genGA+ri and outperforms genGA in the second half. However, does this analysis reflect the real behaviour of the analyzed algorithms? We do not think so. The approach is insufficient, as it seems impossible to draw a satisfactory conclusion from this data.

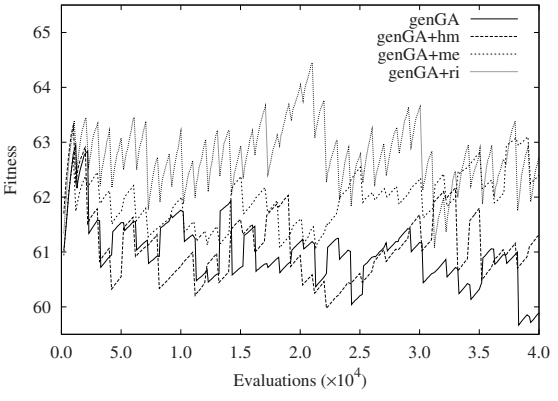


Fig. 2. Mean trace of four algorithms on the moving peaks problem

4.2 Studying the Collective Mean Fitness

Collective mean fitness is a metric with several advantages. First, it does not need to know when changes happen in the environment (as offline performance and error measures need). Second, the value is obtained from the fitness function, which is directly related to the problem. However, Morrison states that F_C should collect information over a representative sample of the fitness landscape dynamics, but there is no clue as what "a representative sample" means. To illustrate this, we will compare the F_C considering three different total run lengths: $40k$, $100k$, and $400k$ evaluations. Numerical results in Table 1 show how much the best F_C values depend on the maximum number of evaluations (stopping condition). We show with these results that even $100k$ evaluations is not useful for a representative sample of the landscape dynamics. However, the main issue here is how can we determine this number and whether a very large value (such as $400k$ evaluations) would provide a representative sample or not.

In order to visually understand this difficulty, we show in Fig. 3 the running F_C for the three considered window spans. Fig. 3a shows that genGA+ri clearly

Table 1. F_C for each algorithm over three different `max_eval` values

Evaluations	genGA	genGA+hm	genGA+me	genGA+ri
40,000	60.9735	61.0878	61.9364	**62.7696**
100,000	60.6000	60.9026	62.4002	**62.8713**
400,000	60.5353	60.8690	**63.2371**	62.8385

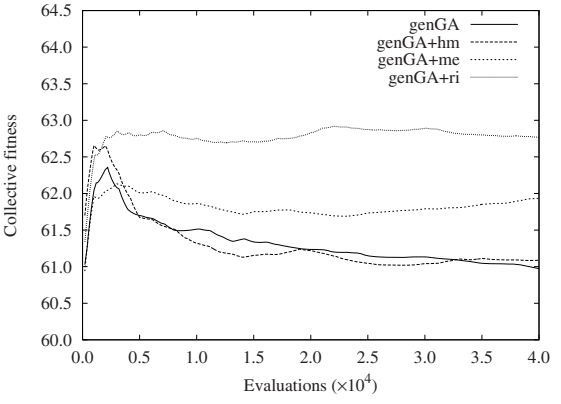
obtains the best results in the $40k$ evaluations case, while genGA+me, genGA, and genGA+hm (from best to worst) are not able to keep its pace. When we extend the simulation to $100k$ evaluations (Fig. 3b), genGA+ri is still pointed as the best algorithm, but genGA+me has significantly improved its performance during the last evaluations. Another change with respect to the shorter experiment is that canonical genGA performance has deteriorated with respect to genGA+hm. To conclude, we show the simulation for the longest considered interval, namely $400k$ evaluations (Fig. 3c). A changing of roles has taken place here, as the trend now shows that genGA+me outperforms genGA+ri, while in the two previous cases it was exactly the contrary conclusion. It seems the algorithms have already been exposed to a significant part of the search landscape, except for genGA+me, whose F_C will improve if the simulation is extended.

In short, in this section we have proved that it is easy to get a given conclusion and its contrary by just allowing the algorithms run a bit further.

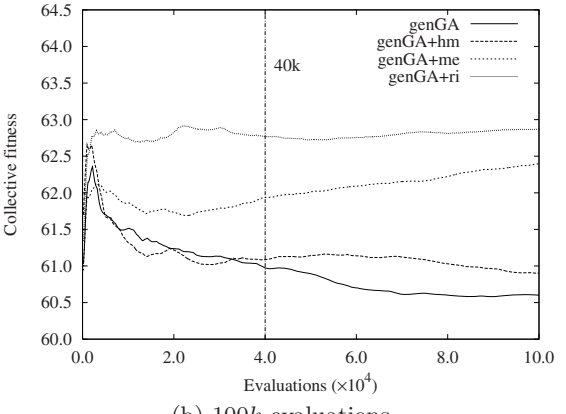
4.3 Studying Weicker's Metrics

This section studies the usage of accuracy, stability, and reactivity to measure the performance of the analyzed algorithms. Among these three measures, accuracy is the main one, while the other two provide complementary results. The advantage of using accuracy is it provides a bounded range of values in $[0, 1]$. However, it is necessary to use a value as reference for the optimum in the present landscape; this value may be unknown in real world problems, an although we could use the current known optimum for such a search space, further research could find new optima and make our results deprecated.

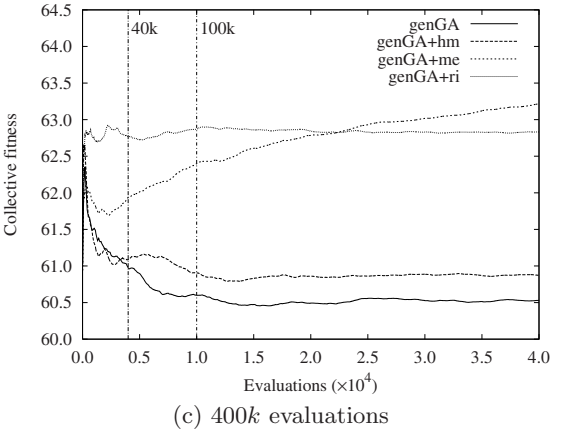
We have compared the results achieved by the three measures and verified that several problems arise. First, accuracy is affected by the same problem as the previously described measures (Table 2a). Second, stability does not directly relate to the goodness of solutions (Table 2b). There, the canonical genGA is the most stable algorithm after $40k$ evaluations, while genGA+me is the most stable one for $100k$ and $400k$ evaluations. Interestingly, the least stable algorithm in all cases is genGA+ri, although it achieves high quality solutions. This is due to high fitness drops when the environment changes. Even if genGA+ri is able to recover fast after changes happen, the severe descent in the accuracy affects the final stability. Finally, as stated in [1], ε-reactivity results are usually unsignificant; in our experiments, all four algorithms obtain the same average ε-reactivity $= 1$.



(a) $40k$ evaluations



(b) $100k$ evaluations



(c) $400k$ evaluations

Fig. 3. Running collective mean fitness for three stopping criteria

Table 2. Weicker's metrics results for $40k$, $100k$, and $400k$ evaluations

(a) Accuracy

Evals	genGA	genGA+hm	genGA+me	genGA+ri
40,000	9.170e-01 $_{6.4e-03}$	9.171e-01 $_{8.7e-03}$	9.338e-01 $_{8.8e-03}$	**9.465e-01** $_{8.9e-03}$
100,000	9.133e-01 $_{9.0e-03}$	9.159e-01 $_{8.7e-03}$	9.396e-01 $_{9.3e-03}$	**9.486e-01** $_{8.5e-03}$
400,000	9.127e-01 $_{9.1e-03}$	9.167e-01 $_{9.1e-03}$	**9.544e-01** $_{1.3e-02}$	9.473e-01 $_{8.6e-03}$

(b) Stability

Evals	genGA	genGA+hm	genGA+me	genGA+ri
40,000	**3.588e-04** $_{1.3e-04}$	3.934e-04 $_{1.2e-04}$	3.901e-04 $_{2.5e-04}$	1.859e-03 $_{1.5e-03}$
100,000	3.852e-04 $_{1.1e-04}$	4.244e-04 $_{1.3e-04}$	**3.707e-04** $_{2.2e-04}$	1.810e-03 $_{1.3e-03}$
400,000	4.010e-04 $_{1.2e-04}$	4.290e-04 $_{1.1e-04}$	**3.234e-04** $_{2.4e-04}$	1.789e-03 $_{1.3e-03}$

5 A Measure for Degradation: $\beta_{degradation}$

None of the previously mentioned metrics account for the progressive degradation suffered by the algorithm. We define degradation as the loss of fitness quality which can affect the optimization process; this results in obtaining worse solutions as time advances. This loss of fitness quality is more obvious and serious when solving DOPs, because it is expected that the algorithm is able to achieve good solutions for a number of different landscapes which follow one another in time. This degradation is more evident as the execution runs for a longer time and affects any of the mentioned existing metrics.

We propose then to measure degradation using linear regression over the consecutive accuracy values achieved at the end of each period. We have selected the accuracy since it is consistently used as the most important error metric in most studies. For that purpose we use Eq. 7, where the variable y is an approximation to the overall accuracy, $\bar{\boldsymbol{x}}$ is a vector of size P, and $\beta_{degradation}$ is the slope of the regression line. P is the number of periods in the dynamic problem. Each x_i is the accuracy of the best solution found in period i averaged over all independent runs N (see Eq. 8). A positive $\beta_{degradation}$ value indicates the algorithm keeps a good improvement and still provides good solutions: the bigger the improvement, the higher the slope value will be. On the contrary, a negative value implies a degradation in the solution quality, where a smaller value implies a deeper loss of quality.

$$y = \beta_{degradation}\, \bar{\boldsymbol{x}} + \varepsilon \tag{7}$$

$$x_i = \frac{1}{N} \sum_{j=1}^{N} f(period\_best_{ji}) \tag{8}$$

The resulting slopes detected in our instances are shown in Table 3. Considering the longest period length $p = 400$, algorithm genGA+me degrades the least; in fact, it is able to improve the accuracy as the optimization process advances.

This trend is already detected with $p = 40$. Canonical genGA is the one most affected by degradation through all experiments. Fig. 4 shows the regression line obtained for the four algorithms. It is also visually evident that genGA+me obtains the ascending line with the steepest slope, which indicates absence of degradation and better improvement of solutions. Besides, genGA obtains the steepest descendant line, which indicates a faster degradation. We can remark that $\beta_{degradation}$ values are of the order of 10^{-4} to 10^{-6}. This is not only due to the high number of periods, but also to accuracy ranging in $[0, 1]$.

Table 3. $\beta_{degradation}$ for each algorithm and number of periods

Periods	genGA	genGA+hm	genGA+me	genGA+ri
40	-4.4835e-04	-1.9975e-04	**8.5789e-05**	-1.9254e-05
100	-2.8144e-05	-1.1124e-04	**1.5504e-04**	3.0762e-05
400	-3.4919e-06	1.5469e-06	**7.3754e-05**	-2.0230e-06

In summary, in this section we have shown that our degradation measure provides representative results for different period lengths, and more important, detects trends in early optimization stages which are later confirmed after longer simulation times.

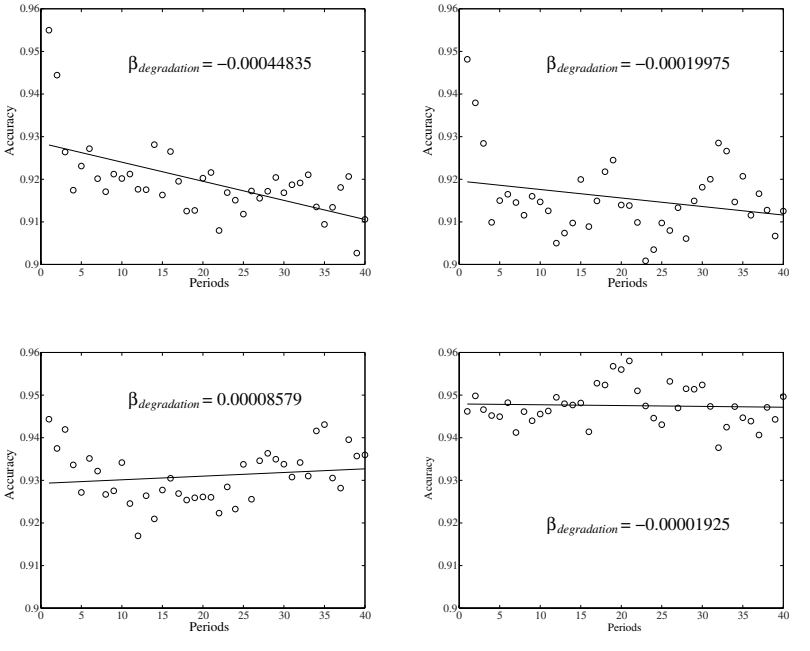


Fig. 4. Linear regression after 40 periods for genGA (top left), genGA+hm (top right), genGA+me (bottom left), and genGA+ri (bottom right)

6 Conclusions

In this article, most popular measures for DOPs have been studied and analyzed on the moving peaks benchmark. We have proposed a new measure for fitness degradation, which we call $\beta_{degradation}$. This measure provides a suitable way of quantifying the loss of solution quality as changes in the landscape occur. It is quantitative, not a mere inspection of a graph, and it is also informative about the behaviour of the algorithm throughout time, unlike accuracy and offline performance. The trend in the performance can be detected earlier than using any of the other measures described in the literature and independently of the simulation time window defined by the researcher.

Future work includes further studies on how fitness degradation affects other DOPs, as well as an extensive study on similar degradation techniques. The normalisation of $\beta_{degradation}$ values in $[0, 1]$ will also be considered.

Acknowledgments

Authors acknowledge funds from the Spanish Ministry of Sciences and Innovation and FEDER under contract TIN2008-06491-C04-01 (M* project) and CICE, Junta de Andalucía under contract P07-TIC-03044 (DIRICOM project).

References

1. Alba, E., Saucedo Badía, J.F., Luque, G.: A study of canonical GAs for NSOPs. In: MIC 2005 Post Conference, ch. 13, pp. 245–260. Springer, Heidelberg (2007)
2. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer, Dordrecht (2001)
3. Bird, S., Li, X.: Informative performance metrics for dynamic optimization problems. In: 9th Conf. on Genetic and Evolutionary Computation, pp. 18–25 (2007)
4. Cobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: 5th Int. Conf. on GAs, pp. 523–530. Morgan Kaufmann, San Francisco (1993)
5. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Parallel Problem Solving from Nature, pp. 137–144. Elsevier, Amsterdam (1992)
6. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. In: 7th International Conference on Genetic Algorithms, pp. 299–306. Morgan Kaufmann, San Francisco (1997)
7. Morrison, R., De Jong, K.: 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)
8. Morrison, R.: Performance measurement in dynamic environments. In: GECCO Workshop on Evolutionary Algorithms for DOPs, pp. 5–8 (2003)
9. Weicker, K.: Performance measures for dynamic environments. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 64–76. Springer, Heidelberg (2002)

Handling Undefined Vectors in Expensive Optimization Problems

Yoel Tenne, Kazuhiro Izui, and Shinji Nishiwaki

Department of Mechanical Engineering and Science-Faculty of Engineering,
Kyoto University, Japan
yoel.tenne@ky3.ecs.kyoto-u.ac.jp,
izui@prec.kyoto-u.ac.jp,
shinji@prec.kyoto-u.ac.jp

Abstract. When using computer simulations in engineering design optimization one often encounters vectors which 'crash' the simulation and so no fitness is associated with them. In this paper we refer to these as *undefined* vectors since the objective function is undefined there. Since each simulation run (a function evaluation) is expensive (anywhere from minutes to weeks of CPU time) only a small number of evaluations are allowed during the entire search and so such undefined vectors pose a risk of consuming a large portion of the optimization 'budget' thus stalling the search. To manage this open issue we propose a *classification-assisted framework for expensive optimization problems*, that is, where candidate vectors are classified in a pre-evaluation stage whether they are defined or not. We describe: a) a baseline single-classifier framework (no undefined vectors in the model) b) a non-classification assisted framework (undefined vectors in the model) and c) an extension of the classifier-assisted framework to a multi-classifier setup. Performance analysis using a test problem of airfoil shape optimization shows: a) the classifier-assisted framework obtains a better solution compared to the non-classification assisted one and b) the classifier can data-mine the accumulated information to provide new insights into the problem being solved.

1 Introduction

Engineering design optimization is a sequential process of improving a system by evaluating and refining candidate designs. Traditionally done in the lab, nowadays researchers use *computer experiments*, that is, simulations which accurately model real-world physics. This setup has two distinct features: a) objective values are obtained from the simulation which is often a legacy software available only as an executable (a *'black-box'*) and this requires gradient-free optimizers and b) each simulation run is *expensive* (requiring anywhere from minutes to weeks of CPU time) and so only a small number of evaluations can be made.

These difficulties are well known and a range of algorithms have been proposed for these optimization problems [9, 13, 14]. However, such optimization problems introduce another difficulty, which has received little attention to date: *the simulation may 'crash' and fail to return an objective value (fitness) for some vectors (candidate designs)*. This

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 582–591, 2010.

has two main implications: a) the objective function is now discontinuous which is problematic for optimizers requiring continuous functions (such as SQP) and b) such vectors can consume a large portion of the optimization budget without improving the fitness landscape and so the optimization search may stall. To the best of our knowledge, no studies have proposed a systematic treatment to handle such undefined vectors in expensive optimization problems, but several studies have acknowledged their existence and the difficulties they introduce, for example [8] mentions 'inputs combinations which are likely to crash the simulator', [10] studies a multidisciplinary problem with 'unevaluable points' which 'cause the simulator to crash', [1] mentions 'virtual constraints' where 'function evaluations fail at certain points'.

In [10] a simple near-neighbour classifier was used to predict which vectors are undefined and assign them a severely penalized fitness (a 'death penalty') with a real-coded evolutionary algorithm but model-assisted optimization was not considered. In a similar penalty-approach [5] re-introduced such vectors into a model but did not use a classifier. We argue such approaches (and similar) are problematic in model-assisted optimization since they: a) discard valuable information on the fitness landscape and b) introduce artificial multimodality to the model (resulting in false optima). In an earlier paper we have proposed handling undefined vectors with a radial basis function model (acting as a classifier) to penalize vectors predicted to be undefined [14]. To the best of our knowledge, this was the first study to consider a classification-assisted optimization framework with models. Later studies have explored the use of classifiers for constrained non-linear programming (but did not focus on handling undefined vectors) [7] (and references therein).

Significantly expanding on [14], the current paper further develops the idea and compares three options for incorporating the 'knowledge' on undefined vectors into the optimization search: a) classifying candidate vectors, using a high penalty and not incorporating undefined vectors in the model b) as above but undefined vectors are assigned a moderate penalty (the mean response of the initial sample) and c) no classification and incorporating undefined vectors directly into the model. We then extend the idea and describe a possible multi-classifier setup. The classification-assisted frameworks we describe have two strong merits: a) *no restrictions are made either on the optimization algorithm nor the type or number of the classifiers* and b) *classification acts as a form of data-mining* and can provide problem insights valuable to end-uses such as engineers and scientists. We analyze the efficacy of the different frameworks using a test problem of airfoil shape optimization where objective values are obtained from a computational aerodynamics simulator.

1.1 The Baseline Algorithm for Expensive Optimization

We first describe the underlying optimization algorithm used in our study, though the proposed framework can be coupled with any other algorithm. Based on our experience and other studies, we use a model-assisted memetic algorithm [9, 13, 14]. The algorithm consists of two main components: a) a model (an interpolant) which approximates the expensive objective function but which is much cheaper to evaluate and b) a memetic algorithm which combines an evolutionary algorithm (EA) and a derivative-free trust-region algorithm, a combination which results in an efficient global–local optimization search.

Due to space constraints we provide only an outline of the algorithm and full details are in [13]. The algorithm begins by generating an initial set of vectors based on a Latin hypercube design of experiments (LHD) [11]. The objective function is evaluated at these vectors and the algorithm then trains an initial model of the objective function. We use a Kriging model which is a statistical approach to interpolation from discrete data [2]. The Kriging model treats the black-box function as a combination of a deterministic function (drift) and a Gaussian random process and model parameters are calibrated using statistical model selection methods such as maximum likelihood.

Next, the optimization algorithm uses an EA to search for an optimum of the model. To improve on the predicted optimum the algorithm then invokes a trust-region derivative-free local-search from the predicted optimum [1]. In contrast to the single model in the global search the local search uses a series of local models (we used local Kriging models but any other model type can be used) to better approximate the function in the trust-region, while the trust-region framework manages the models and ensures the progress of the optimization search. After each optimization iteration (EA search followed by a local search) all new vectors evaluated with the expensive simulation and their corresponding fitness are cached and the process repeats until the number of expensive function evaluations has been exhausted. Algorithm 1 gives a pseudocode of the optimization algorithm.

Algorithm 1. The optimization algorithm

```
generate an initial sample (Latin hypercube design–LHD);
evaluate design vectors and cache;
while fe ⩽ fe_max do
    train a model using the cache;
    use an EA to find an optimum of the model;
    use a trust-region local search from the predicted optimum;
    cache all vectors evaluated with the expensive simulation (and corresponding fitness);
Output: best solution and response found
```

2 Handling Undefined Vectors

This section describes two complementary approaches for handling undefined vectors: classification-assisted and penalty-based.

2.1 Classification-Assisted Optimization

To handle undefined vectors (and as mentioned in Section 1) we introduce a *pre-evaluation stage* into the optimization process, where candidate solutions are first classified as either defined or not by a *classifier*, that is, a function which maps vectors into discrete sets [4].

A wide range of classifiers exist and are discussed in length in various texts [4]. Due to space constraints we can only provide a brief description of the two classifiers which we have experimented with in this study. The first is the simple and robust *nearest-neighbour*

classifier, which works as follows: given a set of training vectors ($\boldsymbol{x}_i, i = 1 \ldots k$) and their respective classes ($y_i = \pm 1$), the classifier checks the distance d (for example l_2) between the new vector ($\boldsymbol{x}_{new}$) to each vector from the training set and assigns the new vector the same class as the closest training vector, namely:

$$y_{new} = y_I : d(\boldsymbol{x}_{new}, \boldsymbol{x}_I) = \min d(\boldsymbol{x}_{new}, \boldsymbol{x}_i)\,,\ i = 1 \ldots k\,. \tag{1}$$

A more recent class, the *support vector machines* (SVM), uses more sophisticated mappings In a two-class problem an SVM tries to find the best classification function for the training data. For a linearly separable training set a linear classification function is the separating hyperplane passing through the middle of the two classes. Once the classifier (hyperplane) is fixed new vectors are classified based on the sign of classifier output (± 1). There are many such hyperplanes so an SVM adds the condition that the function (hyperplane) maximizes the margin between the two classes, geometrically, the distance between the hyperplane and the nearest vectors to it from each class, by maximizing the following Lagrangian:

$$L_P = \frac{1}{2}\|\boldsymbol{w}\| - \sum_{i=1}^{K} \alpha_i y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) + \sum_{i=1}^{K} \alpha_i \tag{2}$$

where $y_i = \pm 1$ is the class of each training vector, $\alpha_i \geqslant 0$ and the derivatives of L_p with respect to α_i are zero. The vector $\boldsymbol{w}$ and scalar b define the hyperplane.

2.2 Technique 1–The 'Explicit' Approach

In this approach the optimization algorithm explicitly classifies candidate vectors as defined or not (hence the name) in a pre-evaluation stage. Vectors classified as defined are assigned the model response, that is, the optimization algorithm receives the same response as if no classifier was present. However, if the vector is classified as undefined then the vector is assigned a penalized (fictitious) response and is returned to the optimizer. In any case, true undefined vectors (those which have crashed the simulator) are not incorporated into the model and so the model (Kriging in our study) is trained using only the defined vectors in the cache. In this setup the pre-evaluation stage is transparent to the underlying optimization algorithm.

An open question is what penalty (fictitious response) should be assigned to vectors classified as undefined. Two apparent options are: a) high penalty (for example, the worst objective value observed in the initial LHD sample) which will cause the the optimizer to quickly discard such vectors and move away from them and b) mean penalty (as above, but the vectors are assigned the mean fitness observed in the initial LHD sample) where the optimizer won't move as quickly from such vectors and will explore their region. In both cases, selecting the penalty based on the objective values from the initial LHD sample ensures the algorithm assigns the same penalty during all stages of the search.

In summary the main features of the explicit technique are: a) uses a classifier to determine what fitness to assign to candidate vectors b) assigns vectors classified as undefined a fixed fictitious fitness (worst or mean response from initial LHD sample) and c) does not incorporate undefined vectors into the model. Algorithm 2 gives a pseudocode of the an optimization framework using the explicit technique.

Algorithm 2. The Explicit Technique

```
generate an initial LHD;
evaluate and cache sites;
find worst or mean sample fitness;
while fe ⩽ fe_max do
    train a model using only the defined vectors in the cache;
    train classifier using all vectors evaluated with the simulation;
    use a classifier to set the predicted response during optimization:
        use and EA to find an optimum of the model;
        evaluate predicted optimum with expensive (true) objective function;
        initiate a local search from the predicted optimum;
        evaluate predicted optimum with expensive (true) objective function;
Output: best solution and response found
```

2.3 Technique 2–The 'Implicit' Approach

As mentioned, a complementary approach is that of assigning undefined vectors a fictitious high penalty and incorporating them into the model, without using any classifier. As such we term the approach 'Implicit' since it does not directly classify candidate vectors but modifies the model response by incorporating highly penalized undefined vectors. Algorithm 3 gives a pseudocode of the an optimization framework using this technique.

Algorithm 3. The Implicit Technique

```
generate an initial LHD;
evaluate and cache sites;
find worst sample fitness (m);
while fe ⩽ fe_max do
    train a model using all vectors in the cache (assign undefined vectors a penalized
    response);
        use and EA to find an optimum of the model
        evaluate predicted optimum with expensive (true) objective function
        initiate a local search from the predicted optimum
        evaluate predicted optimum with expensive (true) objective function
Output: best solution and response found
```

An open question is how effective is this approach since incorporating the highly-penalized undefined vectors will result in an oscillatory landscape which: a) will likely degrade the model accuracy and b) may even introduce false optima into the model landscape. To avoid over-deformation of the model, and for consistency with the Explicit method described in the previous section, we assign undefined vectors the worst fitness observed in the initial LHD sample.

2.4 The Explicit Technique–Extension to Multi-classifier Setups

In this section we describe an extension of the explicit technique to a multi-classifier setup. Specifically, given a user-selected set of classifiers (for example nearest-neighbour, SVM etc.) the technique selects a 'winning' classification output based on

each classifier's past performance. The idea is to record the number of correct classifications each classifier had in a previous 'time-window' of say 10 expensive function evaluations. For a new candidate vector we perform 3 steps: 1) sum the number of correct classifications (in the previous time-window) for all classifiers currently classifying the vector as defined, giving a merit value D 2) as above but for classifiers currently classifying the vector as undefined, giving a merit value U 3) classify the vector as defined if $D > U$, undefined is $D < U$ or select at random if $D = U$.

We can also adjust the fitness assigned to the vector to reflect uncertainty in the correctness of the classification (given by inconsistencies between the classifiers output). Specifically, if all classifiers give the same output (either D or U is zero) we can assume a high degree of certainty than if classifiers give different outputs. To do so we use the above indicators (D and U) in a linear interpolation between the predicted (model) response $\mathcal{S}(\boldsymbol{x})$ and the penalized fitness (worse or mean response of the initial sample) p such that the adjusted fitness of a candidate vector is

$$S_a(\boldsymbol{x}) = \frac{D}{D+U}\mathcal{S}(\boldsymbol{x}) + \frac{U}{D+U}p\,, \tag{3}$$

and denoting $\alpha = D/(D+U)$

$$S_a(\boldsymbol{x}) = \alpha\mathcal{S}(\boldsymbol{x}) + (1-\alpha)p\,. \tag{4}$$

Thus the assigned fitness will be closer to the model prediction if the better-performing classifiers predict the vector is defined but else it will approach the penalized fitness (worst or mean in initial LHD sample). In summary the main features of this extension are: a) uses multiple classifiers and b) assigns a variable fictitious fitness.

3 Performance Analysis

We test the efficacy of three candidate techniques ('explicit' with mean/high penalty and 'implicit') on a test problem of airfoil shape optimization. It is an efficient and effective problem since it captures many of the features of real-world design optimization problems while the simulator run is not prohibitively expensive. The goal is to find an airfoil shape which maximizes the lift coefficient c_{L} and minimizes the drag coefficient (aerodynamic friction) c_{D} at the prescribed flight conditions. Also, the airfoil maximum thickness must be equal to or larger than a critical value ($t^\star = 0.1$, normalized by the airfoil chord) to ensure the airfoil does not break during flight. Figure 1 shows an example. In this study we optimize the airfoil of an aircraft cruising at $30,000\,\mathrm{ft}$, a Mach number $\mathrm{M} = 0.7$ (that is, 70% of the speed of sound) and an angle of attack $\alpha = 10^\circ$ (chosen over lower angles since this increases the simulator crash rate). We used the objective function (a minimization problem)

$$f = -\frac{c_{\mathrm{L}}}{c_{\mathrm{D}}} + \left|\frac{c_{\mathrm{L}}}{c_{\mathrm{D}}}\right| \cdot \frac{\max\{t^\star - t\,,\,0\}}{t^\star} \tag{5}$$

which penalizes airfoils violating the minimum thickness constraint.

To generate candidate airfoils we used the PARSEC parametrization which uses 11 design variables representing geometrical features [12], as Figure 2 shows. We set the bounds on the variables based on our experience as well as previous studies [13]. To obtain the lift and drag coefficients of each candidate airfoil we used XFoil, a computational fluid dynamics simulation which uses the panel method for analysis of subsonic isolated airfoils [3]. Each airfoil evaluation required approximately 30 seconds on a desktop computer. The optimization algorithm considered a simulation run as 'crashed' if: a) it terminated before the 30 seconds limit without producing the lift and drag data (an empty output file) or b) the simulator was still running after 30 seconds (divergence of the numerical solution). We set the limit of expensive function evaluations (simulator calls) to $fe_{\max} = 100$. For the explicit method we used a support vector machine classifier with Gaussian kernels.

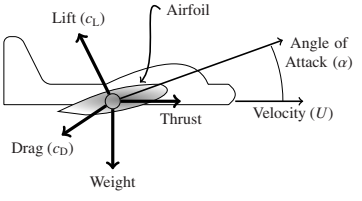


Fig. 1. Lift, Drag, and angle-of-attack (α)

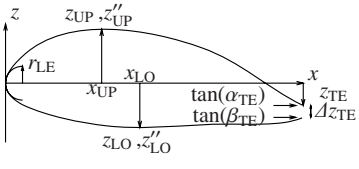


Fig. 2. PARSEC design variables

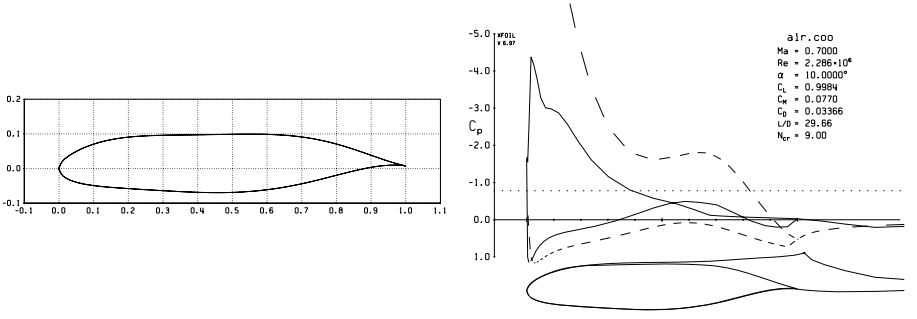
Table 1. PARSEC design variables and their bounds

Variable	Meaning[1]	min.	max.
r_{LE}	LE radius	0.002	0.030
x_{up}	upper thickness (x)	0.2	0.7
z_{up}	upper thickness (z)	0.08	0.18
z''_{up}	upper curvature	−0.6	0.0
x_{lo}	lower thickness (x)	0.2	0.6
z_{lo}	upper thickness (z)	−0.09	0.02
z''_{lo}	lower curvature	0.2	0.9
z_{TE}	TE (z)	−0.01	0.01
Δz_{TE}	TE gap	0	0
α_{TE}[2]	upper TE angle°	165	180
β_{TE}[2,3]	lower TE angle°	165	190

[1] LE: Leading Edge, TE: Trailing Edge
[2] anti-clockwise from the x-axis.
[3] $\beta_{TE} \geqslant \alpha_{TE}$ to avoid intersecting curves.

Figure 3 shows an airfoil found by using the 'explicit technique' with high penalty (Section 2.2) and the variation of the pressure coefficient (c_P) along the upper and lower airfoil curves. The airfoil yields a lift coefficient of $c_L = 0.998$ and a drag coefficient $c_D = 0.033$ and satisfies the thickness requirement (maximum thickness is $t = 0.168$).

We have also benchmarked the 'explicit' technique (with penalty set as either the worst or mean response from the initial LHD sample) against the 'implicit' technique. We repeated tests for 30 times with each algorithm and compared the results using the non-parametric Mann–Whitney test to check if performance gains are statistically-significant (that is, we can reject the null-hypothesis H_0 that the algorithms have performed equivalently) [11]. Table 2 shows the test results from which it follows that the explicit method with higher penalty (worst response) performed best, followed by the explicit with mean penalty. The Mann-Whitney test shows the former's performance



(a) Airfoil geometry (b) Pressure coefficient distribution

Fig. 3. An airfoil obtained by the proposed 'explicit technique' with high penalty: (a) airfoil geometry (b) pressure coefficient distribution along the upper and lower airfoil surfaces

gains are statistically-significant at the $\alpha = 0.05$ level but not at the 0.01 level, and the latter's gains are not statistically-significant. The extent of performance improvement gained by using a classifier will likely depend on how the undefined vectors are distributed with respect to the defined ones in the specific problem being solved. In any case, the classification-assisted framework (explicit technique) performed equally or better than the non-classification assisted one (Implicit technique).

Table 2. Results for the airfoil shape optimization

	Explicit		
Statistic [1,2]	Worst	Mean	Implicit
max	$-1.942e+01$	$-1.744e+01$	$-1.170e+01$
min	$-2.966e+01$	$-3.294e+01$	$-2.704e+01$
mean	$-2.390e+01$	$-2.358e+01$	$-1.979e+01$
SD	$2.804e+00$	$3.533e+00$	$8.331e+00$
median	$-2.401e+01$	$-2.385e+01$	$-2.346e+01$
U (M-W)	$1.863e+00$	$1.478e+00$	

[1] S.D. :standard deviation

[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

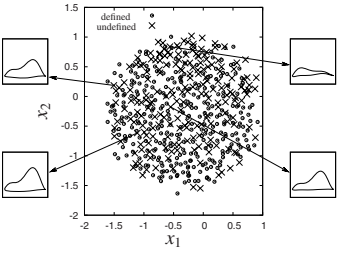


Fig. 4. A Sammon mapping of 500 LHD vectors classified by an SVM classifier. The mapping projects the 11-dimensional vectors onto a two-dimensional map (scatter plot).

As mentioned in Section 1 classification also acts as a form of data-mining and can allow greater insight into the optimization problem. To demonstrate this we used the vectors evaluated during the optimization search to train an SVM classifier (Gaussian kernels) and to predict where undefined vectors exist in the search space. In this example we are interested in: a) approximating how 'dense' is the search space with undefined vectors and b) understanding what causes some vectors to crash the simulation. We trained the classifier using the vectors evaluated during the optimization search and

then used it to classify a LHD sample of 500 vectors. To visualize the distribution of defined and undefined vectors we projected the 11-dimensional vectors (Table 1) into a two dimensional scatter plot using the dimensionality-reduction Sammon mapping [6]. Figure 4 shows the resultant map. Examining the vectors classified as undefined shows they mainly correspond to irregularly shaped airfoils such as those having two or more peaks ('camel humps') unveiling the simulation code (Xfoil) cannot handle such geometries. As such, *the classification-based data-mining allows us to recover a 'virtual-constraint'* (using the terminology from [1]), that is, a constraint induced by the simulation code and which is not part of the original physical problem. Using this insight we can now refine the optimization problem by adjusting the variable bounds to minimize the occurrence of such 'camel humps' airfoils. Overall, both the benchmark tests and the data-mining example show the efficacy and merit of the proposed classification-assisted frameworks.

4 Summary

Real-world expensive optimization problems using computer simulations often stumble upon 'undefined' vectors (candidate solutions) which 'crash' the simulation and can stall the optimization search. To address this we have proposed a classification-assisted framework which assigns a response (fitness) to candidate vectors based on a classifier prediction if they are 'defined' or not (without incorporating undefined vectors into the model), an approach we termed 'Explicit'. A complementary approach (which we termed 'Implicit') assigns undefined vectors a penalized fitness and incorporates them into the model but does not use a classifier during the search. Performance analysis using a test problem of airfoil shape optimization shows that: a) using a classifier can improve the search and result in a better final solution and b) classifiers can data-mine information accumulated during the search thus serving as an additional analysis tool.

References

[1] Conn, A.R., Scheinberg, K., Toint, P.L.: A derivative free optimization algorithm in practice. In: Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization. American Institute of Aeronautics and Astronautics, Reston, Virginia (1998); AIAA Paper AIAA-1998-4718

[2] Cressie, N.A.C.: Statistics for Spatial Data. Wiley, New York (1993)

[3] Drela, M., Youngren, H.: XFOIL 6.9 User Primer. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA (2001)

[4] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, Chichester (2001)

[5] Emmerich, M., Giotis, A., Özedmir, M., Bäck, T., Giannakoglou, K.C.: Metamodel-assisted evolution strategies. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 361–370. Springer, Heidelberg (2002)

[6] Flexer, A.: On the use of self-organizing maps for clustering and visualization. Intelligent Data Analysis 5(5), 373–384 (2001)

[7] Handoko, S., Kwoh, C.K., Ong, Y.S.: Feasibility structure modeling: An effective chaperon for constrained memetic algorithms. IEEE Transactions on Evolutionary Computation (In Print)
[8] Koehler, J.R., Owen, A.B.: Computer experiments. In: Ghosh, S., Rao, C.R., Krishnaiah, P.R. (eds.) Handbook of Statistics, pp. 261–308. Elsevier, Amsterdam (1996)
[9] Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
[10] Rasheed, K., Hirsh, H., Gelsey, A.: A genetic algorithm for continuous design space search. Artificial Intelligence in Engineering 11, 295–305 (1997)
[11] Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 4th edn. Chapman and Hall, Boca Raton (2007)
[12] Sobieckzy, H.: Parametric airfoils and wings. In: Fujii, K., Dulikravich, G.S., Takanashi, S. (eds.) Recent Development of Aerodynamic Design Methodologies: Inverse Design and Optimization, Vieweg, Braunschweig, Wiesbaden. Notes on Numerical Fluid Mechanics, vol. 68, pp. 71–88 (1999)
[13] Tenne, Y.: A model-assisted memetic algorithm for expensive optimization problems. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimisation. Studies in Computational Intelligence, vol. 193. Springer, Heidelberg (2009)
[14] Tenne, Y., Armfield, S.W.: A versatile surrogate-assisted memetic algorithm for optimization of computationally expensive functions and its engineering applications. In: Yang, A., Shan, Y., Thu Bui, L. (eds.) Success in Evolutionary Computation. Studies in Computational Intelligence, vol. 92, pp. 43–72. Springer, Heidelberg (2008)

Adaptive Noisy Optimization

Philippe Rolet and Olivier Teytaud

TAO (Inria), Lri, Cnrs Umr 8623, Univ. Paris-Sud, F-91405 Orsay, France

Abstract. In this paper, adaptive noisy optimization on variants of the noisy sphere model is considered, i.e. optimization in which the *same* algorithm is able to adapt to several frameworks, including some for which no bound has never been derived. Incidentally, bounds derived by [16] for noise quickly decreasing to zero around the optimum are extended to the more general case of a positively lower-bounded noise thanks to a careful use of Bernstein bounds (using empirical estimates of the variance) instead of Chernoff-like variants.

1 Introduction

Noisy optimization is a critical part of optimization since many real-world applications are noisy. It is sometimes called "stochastic optimization" [17,13,6,14], but "stochastic optimization" now often refers to the optimization of deterministic fitness functions by stochastic algorithms. Therefore we will here use "noisy optimization". Noisy optimization often distinguishes between (i) cases in which the variance of the noise quickly decreases to zero around the optimum and (ii) cases in which the variance of the noise is lower bounded. In the literature, various theoretical analyses of complexity bounds can be found for (i), while works covering (ii) are scarce. This paper is concerned with an algorithm covering both frameworks. Various works [8,9,1] have investigated noisy optimization from a theoretical point of view, often with a rough mathematical analysis and sometimes with rigorous arguments (as e.g. [12,16]). In particular, some recent papers investigated the use of *bandit algorithms* [10], inspired from the multi-armed bandit framework (see e.g. [2]), that rely on concentration inequalities such as Hoeffding confidence bounds. The following work proposes a rigorous runtime analysis of noisy expensive optimization based on such a bandit algorithm, in frameworks that are not covered by previously published papers. Specifically, it is shown that the same algorithm can be optimal (within logarithmic factors) for several families of fitness functions simultaneously, extending results of [16] to a quite broader class of noisy fitnesses. The paper is organized as follows. Section 2 defines the framework and introduces some notations. Section 3 states lower bounds that have been derived for this framework in the extant literature, and briefly discusses possible expectations. Section 4 presents *races*, a central tool for our subsequent results. Section 5 introduces the algorithm and proves an upper bound on its runtime. Section 6 provides straightforward extensions of this analysis and pinpoints possible further work.

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 592–601, 2010.

2 Framework

The *black-box* optimization framework is described in Algorithm 1 (similar to [16], but for wider families of fitness functions): the algorithm can request fitness values at any point of the domain, and no other information on the fitness function is available. The paper is interested in *expensive* optimization: it is assumed that obtaining a fitness value is much more costly than running the optimization algorithm. Therefore, the complexity is measured by the number of requests to the fitness. Let X be a domain, and $f : X \times X \to [0, \infty[$ such that $\forall (x, x^*) \in X^2, f(x, x^*) > f(x^*, x^*)$. The class of fitness functions we consider is $\{x \mapsto f(x, t) | t \in X\}$, thus each fitness $f(., t)$ is parameterized by the (unknown) location of its optimum, t. The goal is to find the optimum t (also referred to as x^* in the sequel) of $f(., t)$, by observing noisy measurements of $f(., t)$ at requested points x_i. In the following, t is not handled stochastically, i.e. the lower bounds are not computed *in expectation* w.r.t. all the possible fitness functions yielded by different values of t. Rather, the worst case on t will be considered. For simplicity, we only deal with deterministic algorithms; the extension to stochastic algorithms is straightforward by including a random seed in the algorithm.

Noisy measurements of the fitness at point x are modeled by a random variable $Rand(x)$ satisfying

$$Rand(x) \in [0, 1], \quad \mathbb{E}[Rand(x)] = f(x, x^*) \tag{1}$$

This noise model fits, among others, the goal of finding for a given algorithm a set of parameters minimizing the probability of failure. It notably raises two issues: (i) since few assumptions on the distribution law are made, one cannot use the fact that the probability mass of $Rand(x)$ is centered on $f(x, x^*)$ (it would be the case in i.e. a gaussian noise model). (ii) it is not limited to values worse than those at the optimum as in previous analyses [12]. Importantly, while [1] emphasized that for many algorithms a residual error remains, ours is truly consistent (i.e. $\|x_n - x^*\| \to_\infty 0$) as shown in Theorem 1 of section 5.

Note that the second equation implies:

$$Var[Rand(x)] \leq \mathbb{E}[Rand(x)]. \tag{2}$$

A simple example (from [16]) is $Rand(x) = 1$ with probability $f(x, x^*)$ and 0 otherwise. It is worth noticing that the algorithm considered for proving convergence and upper bound on convergence rates is invariant by addition of a constant. Therefore, our analysis is not restricted to $Rand(x) \in [0, 1]$ since Eq. 2 could be adapted to $Var[Rand(x)] \leq f(x) - \inf_u \inf Rand(u)$ (inf here stands for "essential infimum"). [16] were interested in the sphere function ($f(x, x^*) = ||x - x^*||$), a special case in which the variance promptly decreases to 0 around the optimum. In the sequel, wider classes of fitness functions will be studied:

Scaled sphere function. $f(x, x^*) = \lambda ||x - x^*||$ (case that might be handled similarly to [16]).

Algorithm 1. Noisy optimization framework. *Opt* is a deterministic optimization algorithm; it takes as input a sequence of visited points and their measured fitness values, and outputs a new point to be visited. Noisy fitness values are noted y_n^t since they depend on the fitness $f(.,t)$'s optimum t . The goal is to find points x of the domain such that $f(x,t)$ is as small as possible. Algorithm *Opt* is successful on target function $f(.,t)$ if $Loss(t, Opt)$ is small.

Parameter: N, number of fitness evaluations
for $n \in [[0, N-1]]$ **do**
  $x_{n+1} = Opt(x_1, \ldots, x_n, y_1^t, \ldots, y_n^t)$
  y_{n+1}^t is a draw of random variable $Rand(x_{n+1})$ (see Eqs 1-2)
end for
$Loss(t, Opt) = f(x_N, t)$

Scaled and translated sphere function: (noted S-T sphere from here on). $f(x, x^*) = \lambda||x - x^*|| + c$ (not covered by [16], and fundamentally harder since the variances does *not* decrease to 0 around the optimum).

Transformed sphere. $f(x, x^*) = g(||x - x^*||)$ for some increasing mapping g from $[0, \infty[$ onto a subset of $[0, 1]$.

We consider, in all these cases, a domain X whose diameter satisfies $\sup_{(x,y)\in X^2} ||x - y|| \leq 1$, and $x^* \in X$, so that these settings are well defined. Straightforward extensions are discussed in section 6. In the paper, $[[a, b]] = [a, b] \bigcap \mathbb{N}$. If $(a, b) \in (\mathbb{R}^D)^2$, then $[a, b] = \{x \in \mathbb{R}^D, \forall i \in [[1, D]], a_i \leq x_i \leq b_i\}$.

3 Lower Bounds

In this section we discuss the lower bounds for each of the three models described above. The goal of the rest of the paper will be to show that these bounds are reached within logarithmic factors. Sections 4 and 5 will describe an algorithm which has these guaranteed results on the models discussed above. Interestingly, the algorithm is the same for all models.

Scaled sphere function. In this setting, a direct consequence of [16] is that the distance between the optimum and its approximation is at best $O(1/n)$ after n iterations (for any algorithm). Theorem 3 (sec. 5) shows that this rate can be reached within logarithmic factors.

S-T sphere. Nothing has been poved in this case, to the best of our knowledge. No formal proofs on the matter will be given here; however, here are some intuitive ideas on the behavior of the lower bound. With the S-T sphere function, the variance can be lower bounded by some positive constant c: $\inf_x Var[Rand(x)] > c > 0$. Therefore, evaluating a point n times leads to a confidence interval on its mean whose length is roughly $\sqrt{c/n}$. As a consequence, the precision for an estimate of fitness with n evaluations cannot be less than $\Theta(1/\sqrt{n})$. Since the precision on the fitness space is linear as a function of the precision in the search space, it is reasonable to believe that $\|x_n^+ - x_n^-\| = \Theta(1/\sqrt{n})$

is the best achievable rate. This rate can be reached by our algorithm, as shown in section 5 (Theorem 2).

Monotonically transformed sphere. If the transformation function g is an arbitrary monotonically increasing function, the problem can be made arbitrarily difficult. Therefore, we will only have to show that we guarantee convergence. This consistency will be proved in next section (Theorem 1).

4 Hoeffding/Bernstein Bounds; Their Application to Races

This section recalls some concentration inequalities necessary to analyze the complexity of the algorithm that will be used to prove upper bounds on convergence rates. These inequalities are aimed at quantifying the discrepancy between an average and an expectation. Here, we focus on bounded random variables. The well-known Hoeffding bounds [11] were the first to generalize bounds on binomial random variables to bounded random variables. For some of our purposes, an improved versions of these bounds accounting for the variance [5,3,4], known as Bernstein's bound, will be required. Writing a detailed survey of Hoeffding, Chernoff and Bernstein's bounds is beyond the scope of this paper. We will only present the Bernstein bound, within its application to *races*. A *race* between two or more random variables aims at distinguishing with high confidence random variables with better expectation from those with worse expectation.

Algorithm 2 presents a *Berstein race* for 3 random variables—it is called a Bernstein race because it makes use of the Bernstein confidence bound. The Bernstein race in this paper will be used to distinguish between points x_i of the domain X, based on random variables $Rand(x_i)$. At the end of the race, $3T$

Algorithm 2. Bernstein race between 3 points. Eq. 3 is Bernstein's inequality for estimating the precision for empirical estimates (see e.g. [7, p124]). $\hat{\sigma}_i$ is the empirical estimate of the standard deviation of point x_i's associated random variable $Rand(x_i)$ (it is 0 in the first iteration, which does not alter the algorithm's correctness). $\hat{f}(x)$ is the average of the fitness measurements at x.

$Bernstein(a_1, a_2, a_3, \delta')$
$T = 0$
repeat
  $T \leftarrow T + 1$
  Evaluate the fitness of points x_1, x_2, x_3 once, *i.e.* evaluate the noisy fitness at each of these points.
  Evaluate the precision:

$$\epsilon_{(T)} = 3 \log \left(\frac{3\pi^2 T^2}{6\delta'} \right) / T + \max_i \hat{\sigma}_i \sqrt{2 \log \left(\frac{3\pi^2 T^2}{6\delta'} \right) / T}. \quad (3)$$

until Two points ($good$,bad) satisfy $\hat{f}(bad) - \hat{f}(good) \geq 2\epsilon$ — **return** $(good, bad)$

evaluations have been performed, therefore T is referred to as the halting time in the sequel. The reason why δ' is used in Alg. 2 as the confidence parameter instead of δ will appear later on. Let us define Δ as

$$\Delta = \sup\{\mathbb{E}Rand(x_1), \mathbb{E}Rand(x_2), \mathbb{E}Rand(x_3)\} - \inf\{\mathbb{E}Rand(x_1), \mathbb{E}Rand(x_2), \mathbb{E}Rand(x_3)\}.$$

It is known [15] that if $\Delta > 0$,

- with probability $1 - \delta'$, the Bernstein race is consistent:

$$\mathbb{E}Rand(good) < \mathbb{E}Rand(bad). \tag{4}$$

- the Bernstein race halts almost surely, and with probability at least $1 - \delta'$, the the halting time T verifies

$$T \leq K \log\left(\frac{1}{\delta' \Delta}\right) / \Delta^2 \quad \text{where } K \text{ is a universal constant.} \tag{5}$$

- if, in addition,

$$\Delta \geq C \sup\{\mathbb{E}Rand(x_1), \mathbb{E}Rand(x_2), \mathbb{E}Rand(x_3)\}, \tag{6}$$

  then the Bernstein race halts almost surely, and with probability at least $1 - \delta'$, the halting time T verifies

$$T \leq K' \log\left(\frac{1}{\delta' \Delta}\right) / \Delta \quad \text{where } K' \text{ depends on } C \text{ only.} \tag{7}$$

The interested reader is referred to [15] and references therein for more.

5 Upper Bounds for Noisy Optimization

Algorithm 3 (based on the Bernstein race discussed above) will be used for proving our upper bounds. This algorithm was proposed in [16], with a weaker version of races. In the present work, the race was improved so that it can deal with more general settings than those of [16]. Informally, the algorithm (adapted from [16] for the case of variance not decreasing to zero around the optimum) is as follows. The domain is a hyper-rectangle $[x_0^-, x_0^+]$ of $\mathbb{R}^D$. Define the *axes* of a hyper-rectangle as the lines parallel to any edge of the hyper-rectangle, and containing the center of the hyper-rectangle. At iteration n, the algorithm considers the axis on which the hyper-rectangle is the largest (any rule for breaking ties is allowed). Three points are placed along this axis, one at the center of the hyper-rectangle, and the two others at the two intersections between the axis and the hyper-rectangle's frontier.

Then, the algorithm uses the Bernstein race for selecting a point $good_n$ and a point bad_n among these three points, such that the $good_n$ point is closer to

Algorithm 3. Algorithm for optimizing noisy fitness functions. *Bernstein* denotes a Bernstein race, as defined in Algorithm 2. The initial domain is $[x_0^-, x_0^+] \in \mathbb{R}^d$. δ is the confidence parameter.

$n \leftarrow 0$
while True **do**
  $c = \arg\max_i (x_n^+)_i - (x_n^-)_i$ // Pick the coordinate with highest uncertainty
  $\delta_n^{\max} = (x_n^+)_c - (x_n^-)_c$
  for $i \in [[1,3]]$ **do**
    ${x'}_n^i \leftarrow \frac{1}{2}(x_n^- + x_n^+)$. // Consider the middle point
    $({x'}_n^i)_c \leftarrow (x_n^-)_c + \frac{i-1}{2}(x_n^+ - x_n^-)_c$. //except that the c^{th} coordinate may take // 3 different values
  end for
  $(good_n, bad_n) = Bernstein({x'}_n^1, {x'}_n^2, {x'}_n^3, \frac{6\delta}{\pi^2(n+1)^2})$. // a good and a bad point
  Let H_n be the halfspace $\{x \in \mathbb{R}^D; ||x - good_n|| \leq ||x - bad_n||\}$.
  Split the domain: $[x_{n+1}^-, x_{n+1}^+] = H_n \cap [x_n^-, x_n^+]$.
  $n \leftarrow n + 1$
end while

the optimum than the bad_n point. The Bernstein race described in section 4 by algorithm 2 guarantees this with confidence $1 - \delta'$.[1]

In the *transformed sphere* models under analysis, $\mathbb{E}\ Rand(x)$ is increasing as a function of $\|x - x^*\|$, thus the optimum is in the hyper-rectangle $H = \{x \in \mathbb{R}^D; \|x - good_n\| \leq \|x - bad_n\|$ with probability $1 - \delta$. The first lemma for our proofs is given below:

Lemma 1. *Let $\delta > 0$, and let fitness f be an increasing transformation of the sphere function $x \mapsto \|x - x^*\|$.*[2] *Let $Rand(x)$ be the noisy answer to an evaluation of f as defined above. If, in algorithm 3, the Bernstein race halts at all steps until iteration n, then:*

$$\left(\frac{3}{4}\right)^n ||x_0^+ - x_0^-|| \leq ||x_n^+ - x_n^-|| \leq \left(\frac{3}{4}\right)^{\lfloor n/D \rfloor} ||x_0^+ - x_0^-||, \tag{8}$$

$$\text{and} \quad (\forall i < n, \mathbb{E}Rand(good_i) \leq \mathbb{E}Rand(bad_i)) \Rightarrow x^* \in [x_n^-, x_n^+], \tag{9}$$

and for some constant K depending on the dimension only,

$$x^* \in [x_n^-, x_n^+] \Rightarrow \exists (good_n, bad_n) \in \{{x'}_n^1, {x'}_n^2, {x'}_n^3\}^2, \tag{10}$$

$$\|x^* - bad_n\| \geq \|x^* - good_n\| + K||x_n^+ - x_n^-|| \tag{11}$$

[1] Note that this particular kind of race is not interested in knowing how good remaining points (other than $good_n$ and bad_n) are. It might be that in our case the third point is even closer to the optimum, but the point of this race is not to determine the closest point, it is to provide two points such that one is closer than the other.

[2] The transformed sphere covers models of the S-T sphere, and of the scaled sphere.

Due to length constraints, the proof of this lemma is not given here. A very similar lemma is used for the case of a variance decreasing to zero around the optimum, in [16]. □

A consequence of this lemma is the following convergence guarantee:

Theorem 1 (Consistency of Algo. 3 for the transformed sphere). *In the transformed sphere model, Algo. 3 ensures* $x_n^- \to x^*$ *and* $x_n^+ \to x^*$ *with probability at least* $1-\delta$.

Proof
Eq. 9 of the previous lemma implies that $\|x_n^+ - x_n^-\| \to 0$. We will now show that with probability $1-\delta$, $x^* \in [x_n^-, x_n^+]$ by establishing the left-hand side of Eq. 4 by induction. This will be sufficient to prove theorem 1.

- The induction hypothesis $\mathcal{H}(n)$ is as follows:

  With probability at least $1 - \sum_{k=1}^{n+1} \frac{6\delta}{\pi^2 k^2}, \forall i < n, \mathbb{E}Rand(good_i) \leq \mathbb{E}Rand(bad_i)$.

- $\mathcal{H}(0) : x^* \in [x_0^-, x_0^+]$ by definition.
- Let us assume $\mathcal{H}(n-1)$ for $n > 0$. For clarity, the statement $\forall i < n, \mathbb{E}Rand(good_i) \leq \mathbb{E}Rand(bad_i)$ is written $G(n)$.

$$\begin{aligned} P(G(n)) &= P(G(n-1), \mathbb{E}Rand(good_n) \leq \mathbb{E}Rand(bad_n)) \\ &= P(\mathbb{E}Rand(good_n) \leq \mathbb{E}Rand(bad_n) \mid G(n-1)) P(G(n-1)) \\ &= (1 - \sum_{k=1}^{n} \frac{6\delta}{\pi^2 k^2})(1 - \frac{6\delta}{\pi^2 (n+1)^2}) \qquad (12) \\ &\geq 1 - \sum_{k=1}^{n+1} \frac{6\delta}{\pi^2 k^2} \end{aligned}$$

  which proves $\mathcal{H}(n)$. The first term of eq. 12 is the application of $\mathcal{H}(n-1)$. The second term is a property of the Bernstein race described in Algo. 2, and used in Algo. 3.

It only remains to observe that $\sum_{i=1}^{\infty} (6\delta/(\pi i)^2) = \delta$ to conclude. □

The number of iterations is of course log-linear ($\log(\|x_n^- - x^*\|)/n$ is upper bounded by a negative constant), but the number of evaluations per iteration might be arbitrary large. More precise (faster) results, for the other (simpler) models will now be considered.

Theorem 2 (Hoeffding rates for the S-T sphere model). *Consider the S-T sphere model, and a fixed dimension* D. *The number of evaluations requested by Algo. 3 for reaching precision* ϵ *with probability at least* $1-\delta$ *is* $O(\frac{\log(\log(1/\epsilon)/\delta)}{\epsilon^2})$.

Proof. Eq. 11 ensures that

$$\begin{aligned}\Delta_n =& \sup\{\mathbb{E}Rand({x'}^1_n), \mathbb{E}Rand({x'}^2_n), \mathbb{E}Rand({x'}^3_n)\} \\ & - \inf\{\mathbb{E}Rand({x'}^1_n), \mathbb{E}Rand({x'}^2_n), \mathbb{E}Rand({x'}^3_n)\} \\ & \geq \|x^* - bad_n\| - \|x^* - good_n\|\end{aligned}$$

verifies $\Delta_n = \Omega(\|x_n^+ - x_n^-\|)$. Therefore, applying the concentration inequality 5, the number of evaluations in the n^{th} iteration is at most

$$O\left(\log\left(\frac{6\delta}{\pi^2(n+1)^2}\right)/\|x_n^- - x_n^+\|^2\right). \tag{13}$$

Now, let us consider the number $N(\epsilon)$ of iterations before a precision ϵ is reached. Eq. 8 shows that there is a constant $k < 1$ such that

$$\epsilon \leq \|x_n^+ - x_n^-\| \leq Ck^{N(\epsilon)} \tag{14}$$

Injecting Eq. 14 in Eq. 13 shows that the cost (the number of evaluations) in the last call to the Bernstein race is

$$Bound_{last}(\epsilon) = O\left(-\log\left(\frac{6\delta}{\pi^2(N(\epsilon)+1)^2}\right)/\epsilon^2\right). \tag{15}$$

Since $N(\epsilon) = O(\log(1/\epsilon))$, $Bound_{last} = O(\log(\log(1/\epsilon')/\delta))$. For a fixed dimension D, there exists $k' > 1$ such that the cost of the $(N(\epsilon) - i)^{th}$ iteration is at most

$$O(\lceil Bound_{last}/(k')^i \rceil) \tag{16}$$

because the algorithm ensures that after D iterations, $\|x_n^+ - x_n^-\|$ decreases by at least $3/4$.

The sum of the costs for $N(\epsilon)$ iterations is therefore the sum of $O(Bound_{last}/(k')^i)$ for $i \in [[0, N(\epsilon) - 1]]$, that is $O(Bound_{last}/(1 - k')) = O(Bound_{last})$ (plus $O(N(\epsilon))$ for the rounding associated to the $\lceil ... \rceil$ in Eq. 16).

The overall cost is therefore $O(Bound_{last} + \log(1/\epsilon))$. This yields the expected result. □

Theorem 3 (Bernstein rates for the scaled sphere model). *Consider the scaled sphere model, and a fixed dimension D. Then, the number of evaluations requested for reaching precision ϵ with probability at least $1-\delta$ is $O(\frac{\log(\log(1/\epsilon)/\delta)}{\epsilon})$.*

Proof. The proof follows the lines of the proof of Theorem 2, except for one point. As well as for Theorem 2, we use the fact that for the scaled sphere model (and in fact also for the S-T sphere model), Eq. 11 holds, which implies (with $\Delta_n = \sup_i \mathbb{E}Rand({x'}^i_n) - \inf_i \mathbb{E}Rand({x'}^i_n)$)

$$\Delta_n = \Omega(\|x_n^+ - x_n^-\|). \tag{17}$$

However, for the scaled sphere model, we can also claim

$$\sup_i \mathbb{E}Rand({x'}^i_n) = O(\|x_n^+ - x_n^-\|). \tag{18}$$

Eqs. 17 and 18 lead to Eq. 6.

Furthermore, Eq. 6 implies that Eq. 15 can be replaced by

$$Bound_{last}(\epsilon) = O\left(-\log\left(\frac{6\delta}{\pi^2(N(\epsilon)+1)^2}\right)/\epsilon\right). \tag{19}$$

The summation as in the proof of Theorem 2 now leads to an overall cost $O(\frac{\log(\log(1/\epsilon)/\delta)}{\epsilon})$. □

6 Discussion

We considered the optimization of noisy fitness functions, where the fitness in x is randomized, with values in $[0,1]$, and expected value $f(x,x^*)$ where x^* is the optimum. The following models were studied: (i) Sphere function: $f(x,x^*) = \|x-x^*\|$; (ii) Scaled sphere function: $f(x,x^*) = \lambda\|x-x^*\|$; (iii) S-T sphere function: $f(x,x^*) = \lambda\|x-x^*\|+c$; (iv) Transformed sphere: $f(x,x^*) = g(\|x-x^*\|)$. The first case only was in the state of the art. The same algorithm (using Bernstein's inequality) ensures that with probability $1-\delta$, the optimum x^* is in a set of diameter δ_n (after n fitness evaluations), which provably decreases as shown in Table 1. There are some straightforward extensions, the main one being that convergence rates only depends on $f(x,x^*)$ for x close to x^*: all f such that $f(x,x^*) = \Theta(||x-x^*||)$ lead to the same asymptotic rate as the scaled sphere; and all f such that $f(x,x^*) - c = \Theta(||x-x^*||)$ lead to the same asymptotic rate as the scaled and translated sphere function. Therefore, it is likely that the proposed approach is much more general than variants of the sphere model as formally considered here. It has been shown in [16] that some links exist between the rates for $f(x,x^*) = ||x-x^*||$ and $f(x,x^*) = ||x-x^*||^p$; these links will not be developed here. The main further works are: (i) formalizing the lower bound for the case of the scaled and translated sphere function; (ii) experiment real-world algorithms or adapted version of real-world algorithms (as e.g. [10]) on these fitness functions.

Table 1. Precision δ_n (diameter of the region in which might be the optimum) as a function of the number n of fitness evaluations. The $\tilde{O}(.)$ means that logarithmic factors are present. Dependencies in δ can be found in detailed results; the dependency in the dimension can be computed from the proofs, but we guess they are not optimal. The constants depend on λ and on the dimension; c has no impact on the constant for the scaled and translated sphere function.

Model	Precision $\|x_n^- - x_n^+\|$
Sphere function	$\tilde{O}(1/n)$
Scaled sphere function	$\tilde{O}(1/n)$
Scaled and translated sphere function	$\tilde{O}(1/\sqrt{n})$
Transformed sphere	$o(1)$

References

1. Arnold, D.V., Beyer, H.-G.: Efficiency and mutation strength adaptation of the (mu/mui,lambda)-es in a noisy environment. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 39–48. Springer, Heidelberg (2000)
2. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research 3, 397–422 (2003)
3. Bernstein, S.: On a modification of chebyshev's inequality and of the error formula of laplace. Original publication: Ann. Sci. Inst. Sav. Ukraine, Sect. Math. 1 3(1), 38–49 (1924)
4. Bernstein, S.: The Theory of Probabilities. Gastehizdat Publishing House, Moscow (1946)
5. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Annals of Math. Stat. 23, 493–509 (1952)
6. Denton, B.: Review of "stochastic optimization: Algorithms and applications" by stanislav uryasev and panos m. Interfaces 33(1), 100–102 (2003)
7. Devroye, L., Györfi, L., Lugosi, G.: A probabilistic Theory of Pattern Recognition. Springer, Heidelberg (1997)
8. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic algorithms in noisy environments. Machine Learning 3, 101–120 (1988)
9. Hammel, U., Bäck, T.: Evolution strategies on noisy functions: How to improve convergence properties. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 159–168. Springer, Heidelberg (1994)
10. Heidrich-Meisner, V., Igel, C.: Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In: ICML 2009: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 401–408. ACM, New York (2009)
11. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58, 13–30 (1963)
12. Jebalia, M., Auger, A.: On multiplicative noise models for stochastic search. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 52–61. Springer, Heidelberg (2008)
13. Kall, P.: Stochastic Linear Programming. Springer, Berlin (1976)
14. Marti, K.: Stochastic Optimization Methods. Springer, Heidelberg (2005)
15. Mnih, V., Szepesvári, C., Audibert, J.-Y.: Empirical Bernstein stopping. In: ICML 2008: Proceedings of the 25th international conference on Machine learning, pp. 672–679. ACM, New York (2008)
16. Rolet, P., Teytaud, O.: Bandit-based estimation of distribution algorithms for noisy optimization: Rigorous runtime analysis. Submitted to Lion4; presented in TRSH 2009 in Birmingham (2009)
17. Sengupta, J.K.: Stochastic Programming. Methods and Applications. North-Holland, Amsterdam (1972)

Noise Analysis Compact Genetic Algorithm*

Ferrante Neri, Ernesto Mininno, and Tommi Kärkkäinen

Department of Mathematical Information Technology P.O. Box 35 (Agora) 40014
University of Jyväskylä, Finland
{ferrante.neri,ernesto.mininno,tommi.karkkainen}@jyu.fi

Abstract. This paper proposes the Noise Analysis compact Genetic Algorithm (NAcGA). This algorithm integrates a noise analysis component within a compact structure. This fact makes the proposed algorithm appealing for those real-world applications characterized by the necessity of a high performance optimizer despite severe hardware limitations. The noise analysis component adaptively assigns the amount of fitness evaluations to be performed in order to distinguish two candidate solutions. In this way, it is assured that computational resources are not wasted and the selection of the most promising solution is correctly performed. The noise analysis employed in this algorithm spouses very well the pair-wise comparison logic typical of compact evolutionary algorithms. Numerical results show that the proposed algorithm significantly improves upon the performance, in noisy environments, of the standard compact genetic algorithm. Two implementation variants based on the elitist strategy have been tested in this studies. It is shown that the nonpersistent strategy is more robust to the noise than the persistent one and therefore its implementation seems to be advisable in noisy environments.

1 Introduction

In several real-world applications, an optimization problem must be solved despite the fact that a full power computing device may be unavailable due to cost and/or space limitations. This situation is typical of robotics and control problems. For example, in order to obtain a portable electric device, the optimization of the control system must be encoded within the memory of the control card since a computer cannot be plugged to the system. In order to overcome this class of problems compact Evolutionary Algorithms (cEAs) have been designed. A cEA is an Evolutionary Algorithm (EA) belonging to the class of Estimation of Distribution Algorithms (EDAs), see [13]. The algorithms belonging to this class do not store and process an entire population and all its individuals therein but on the contrary make use of a statistic representation of the population in order to perform the optimization process. In this way, a much smaller amount of parameters must be stored in the memory. Thus, a run of these algorithms requires much less capacious memory devices compared to their correspondent

* This research is supported by the Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing and by Tekes - the Finnish Funding Agency for Technology and Innovation, grant 40214/08 (Dynergia).

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 602–611, 2010.

standard EAs. The first cEA was the compact Genetic Algorithm (cGA) introduced in [11]. The cGA simulates the behavior of a standard binary encoded standard Genetic Algorithm (GA). Paper [1] analyzes analogies and differences between cGAs and $(1+1)$-ES and extends a mathematical model of ES [18] to cGA obtaining useful information on the performance. Moreover, [1] introduces the concept of elitism, and proposes two new variants, with strong and weak elitism respectively, that significantly outperform both the original cGA and $(1+1)$-ES. A real-encoded cGA (rcGA) has been introduced in [14]. Some examples of rcGA applications to control engineering are given in [7] and [8].

Due to the presence of approximation models, neural networks, and measurement systems the optimization cost function is likely to be noisy. Under these conditions, some countermeasure must be designed in order to perform the optimization despite the presence of noise. Due to their inner structure and population based setup, Evolutionary Algorithms (EAs) are considered to be very promising with noisy problems and manage to improve upon initial fitness values, see [3], and [2]. However, the noise still represents a challenge that needs to be addressed, and standard EAs often do not manage to detect satisfactory solutions. In particular, the most critical operation is, as highlighted in [4], selection of the most promising solutions for subsequent phases of the optimization algorithms. In order to overcome this difficulty, as highlighted in [12], an averaging operation must be implemented to filter the noise and allow a correct selection of the most promising solution. This fact leads to an unavoidable increase of the computational cost due to extra samplings and thus fitness evaluations. Thus, several papers propose methods for handling the noise without an excessive increase in the computational cost. Some examples of adaptive sampling strategies are given in [20] and [16]. In other papers, e.g. [5], sequential approaches which aim at reducing the sample size during the tournament selection and performing a massive re-sampling only when strictly necessary have been proposed. Recently, several heuristics for improving the robustness to noise have been introduced in the context of multi-objective optimization, see [10]. In [17] and [15] a memetic approach is proposed to handle the uncertainties due to the presence of noise. In [6], a noise analysis component has been proposed for the survivor selection scheme of Differential Evolution.

If the application requires the solution of an optimization problem despite both limited hardware conditions and the presence of noise, a population based approach is not applicable and the employment of classical compact algorithms (as well as classical population based algorithms) might lead to unsatisfactory results due to the pernicious effect of noise. In this paper, we propose the first (to our knowledge) implementation of compact algorithm integrating a noise handling component, see [12]. This algorithm employs a compact genetic algorithm based on the study in [14] and a noise analysis structure for efficiently selecting the elite individuals. The remainder of this paper is organized in the following way. Section 2 describes the features of the problem under consideration and describes the proposed algorithm. Section 3 gives the experimental setup and numerical results of the present study. Section 4 gives the conclusions of this paper.

2 Noise Analysis Compact Genetic Algorithm

A stationary optimization problem must be reformulated in the case of a noisy problem. As shown in [12], a noisy fitness function $\tilde{f}(x)$ (affected by Gaussian noise) can mathematically be expressed as:

$$\int_{-\infty}^{\infty} [f(x) + z]\, p(z)\, dz = f(x), z \sim N(0, \sigma^2) \tag{1}$$

where x is the design vector, $f(x)$ is a time-invariant function, z is an additive noise normally distributed with 0 mean and variance σ^2, and $p(z)$ is the distribution function of the noise.

In principle, search of the optimum consists of optimizing $f(x)$; however since only the fitness values related to the $\tilde{f}(x)$ are available, the noisy optimization problem consists of optimizing $\tilde{f}(x)$ in a decision space D where x is defined. Without a loss in generality, this paper will refer to minimization problems.

In order to minimize the fitness function in formula (1) for those applications which do not allow a full power computational device, the Noise Analysis compact Genetic Algorithm (NAcGA) has been designed. The NAcGA consists of the following steps. At the beginning of the optimization process, a Probability Vector (PV) is initialized. The PV is a $n \times 2$ matrix, see [14]:

$$PV^t = \left[\mu^t, \Sigma^t\right] \tag{2}$$

where n is the amount of design variable of the problem, μ and Σ are, respectively, vectors containing, for each design variable, mean and standard deviation values of a Gaussian Probability Distribution Function (PDF) truncated within the interval $[-1, 1]$. The height of the PDF has been normalized in order to keep its area equal to 1. The apex t indicates the generation (number of performed comparison number).

During the initialization, for each design variable i, $\mu^1[i] = 0$ and $\Sigma^1[i] = \lambda$ where λ is a large positive constant (e.g. $\lambda = 10$). This initialization of Σ values is done in order to simulate a uniform distribution. Subsequently, one individual is sampled as elite. A new individual is generated and compared with the elite. In order to efficiently perform the fitness comparison despite the noise, the noise analysis component proposed in [6] has been adapted and integrated into the compact structure. In order to explain its working principle, let us indicate with x_1 the elite and x_2 the newly generated individual. In order to compare their fitness scores the the value $\delta = \left|\bar{f}(x_1) - \bar{f}(x_2)\right|$ is computed. With $\bar{f}$ we mean the average fitness calculated over n_s samples. If $\delta > 2\sigma$ the candidate solution displaying the best performance value is simply chosen for the subsequent generation. This choice can be justified considering that, for a given Gaussian distribution, 95.4% of the samples fall in an interval whose width is 4σ and has at its center the mean value of the distribution, see [19]. In this case, if the difference between two fitness values is greater than 2σ, it is likely

that the point which seems to have a better fitness is truly the best performing of the two candidate solutions.

On the other hand, if $\delta < 2\sigma$, the noise bands related to the two candidate solutions do overlap, and determining a ranking based on only one fitness evaluation is impossible. In this case, indicating with $\alpha = min\left\{\bar{f}\left(x_1\right), \bar{f}\left(x_2\right)\right\}$ and $\beta = max\left\{\bar{f}(x_1), \bar{f}\left(x_2\right)\right\}$, the following index is calculated:

$$\upsilon = \frac{\alpha + 2\sigma - (\beta - 2\sigma)}{\beta + 2\sigma - (\alpha - 2\sigma)}. \tag{3}$$

The index υ represents the intersection of two intervals, characterized by a center in the fitness value and semi-width 2σ, with respect to their union. In other words, υ is a normalized measure of the noise band overlap. This index can vary between 0 and 1. The limit condition $\upsilon \approx 0$ means that overlap is limited and thus the pairwise ranking given by the single sample estimations is most likely correct. The complementary limit condition, $\upsilon \approx 1$ means that the interval overlap is almost complete and the two fitness values are too close to be distinguished in the noisy environment. In other words, υ can be seen as a reliability measure of a pairwise solution ranking in the presence of noisy fitness.

On the basis of the calculated value of υ a set of additional samples n_s is performed for both the solutions x_1 and x_2; their respective fitness values $\bar{f}_i$ are then updated by averaging over the performed samples. These samples are determined by calculating:

$$n_s = \left\lceil \left(\frac{1.96}{2 \cdot (1 - \upsilon)} \right)^2 \right\rceil, \tag{4}$$

where 1.96 is the upper critical value of a normal distribution associated to a confidence level equal to 0.95, see [19].

Thus, n_s represents the minimum amount of samples which ensure a reliable characterization of the noise distribution, i.e., the amount of samples which allows consideration of the average fitness values as the mean value of a distribution.

To better understand how formula (4) has been derived, let us consider a set of normally distributed samples $(a_1, a_2, \ldots, a_n)$. The theoretical average μ of the stochastic process associated to the samples is the center of a confidence interval

$$\bar{a} - \gamma \leq \mu \leq \bar{a} + \gamma, \tag{5}$$

where $\bar{a}$ is the average of the given samples and γ is the semi-amplitude of the confidence interval. In order to be sure that 95% of the samples fall within the confidence interval in inequalities (5), the semi-amplitude γ should be:

$$\gamma = \frac{\sigma}{\sqrt{n_s}} z, \tag{6}$$

where σ is the standard deviation of the stochastic process, n_s is the amount of samples, and z is the upper critical value from the normal distribution. For a

confidence level equal to 0.95, the upper critical value is 1.96. Thus, the minimum amount of samples for characterizing a Gaussian process with a confidence level of 0.95 is:

$$n_s = \left(\frac{1.96}{\gamma}\right)^2 \sigma^2. \tag{7}$$

Then n_s fitness evaluations are performed on both x_1 and x_2 and their corresponding average fitness values $\bar{f}$ are computed. It is clear that for $\gamma \approx 2\sigma$ one sample is enough to estimate the center of the distribution. Since the problem of the ranking of two fitness values corresponds to the correct estimation of the centers of two distributions, then if the fitness estimations are more distant than 4σ ($\upsilon \approx 0$) one sample is enough to distinguish two solutions. On the contrary, it has been imposed that the amount of samples to be performed for each solution hyperbolically increases with the overlapping of the two distribution estimations.

However, as shown in eq. (4), since for $\upsilon \to 1$ it would result in $n_s \to \infty$, a saturation value for n_s (i.e., the maximum amount of samples for each solution) has been set in order to avoid infinite loops. It must be remarked that this saturation value is the only extra parameter to be set, with respect to a standard DE. In addition, setting of this parameter can be intuitively carried out on the basis of the global computational budget available and the precision requirement in the specific application.

It must be noticed that the noise analysis procedure proposed in [6] can be applied only when the noise is supposed to be Gaussian and the standard deviation value (or its estimation) is known in advance. However, many noisy optimization problems are actually characterized by Gaussian noise, due to the presence of measurement errors. In addition, if the noise cannot be approximated by a Gaussian distribution but by a different known distribution, the procedure can still be modified and adapted to the specific case. Finally, the standard deviation in many cases can be easily estimated by performing a preliminary analysis of noise. In the case of noise due to measurement errors, the standard deviation can be derived from the data plate of the measurement devices.

On the basis of the average fitness values, the old elite is retained or replaced by the newly generated solution. When this operation is performed the PV is updated following the rules given in [14]. The update rule for μ values is given by:

$$\mu^{t+1} = \mu^t + \frac{1}{N_p}\left(winner - loser\right), \tag{8}$$

where N_p is virtual population size. The update rule for σ values is given by:

$$\left(\Sigma^{t+1}\right)^2 = \left(\Sigma^t\right)^2 + \left(\mu^t\right)^2 - \left(\mu^{t+1}\right)^2 + \frac{1}{N_p}\left(winner^2 - loser^2\right). \tag{9}$$

With the terms *winner* and *looser* we refer to the individuals with higher and lower fitness performance, respectively. Details for constructing formulas (8) and (9) are given in [14].

In addition to this algorithmic structure, two elitist schemes have been tested in this paper in the fashion of persistent and nonpersistent elitism, proposed in [1]. Thus, two implementations of NAcGA are proposed in this paper. In the

counter $t = 0$ and $\theta = 0$
initialize PV
generate *elite* x_1 by means of PV
while budget condition **do**
  generate 1 individual x_2 of by means of PV
  {** Noise Analysis **}
  calculate $\upsilon = \frac{\alpha+2\sigma-(\beta-2\sigma)}{\beta+2\sigma-(\alpha-2\sigma)}$
  calculate $n_s = \left\lceil \left(\frac{1.96}{2\cdot(1-\upsilon)} \right)^2 \right\rceil$
  calculate fitness values of x_1 and x_2 n_s times and their average fitness values $\bar{f}$
  {** Elite Selection **}
  $[winner, loser] = compete\,(a, elite)$
  if $x_2 == winner$ OR $\theta \geq \eta$ **then**
    $elite = x_2$
    $\theta = 0$
  end if
  {** PV Update **}
  $\mu^{t+1} = \mu^t + \frac{1}{N_p}\,(winner - loser)$
  $\left(\Sigma^{t+1}\right)^2 = \left(\Sigma^t\right)^2 + \left(\mu^t\right)^2 - \left(\mu^{t+1}\right)^2 + \frac{1}{N_p}\left(winner^2 - loser^2\right)$
  counter update $t = t + 1$
end while

Fig. 1. ne-NAcGA pseudo-code

persistent elitism Noise Analysis compact Genetic Algorithm (pe-NAcGA) the newly generated solution replaces the elite only when it outperforms it. In the nonpersistent elitism Noise Analysis compact Genetic Algorithm (ne-NAcGA), the replacement occurs either when the newly generated solution outperforms the elite or when the elite has not been replaced after η comparisons. Several comparisons about which elitist scheme is preferable have been carried out. In general, whether the persistent or nonpersistent scheme is preferable seems to be a problem dependent issue, see [1]. For the sake of clarity, the pseudo-code displaying the working principles of ne-NAcGA is given in Fig. 1. The persistent elitist version has a very similar pseudo-code but there is not the "IF" condition on η. With the function *compete* we simply mean (average) fitness comparison in order to detect the individual with the best performance.

3 Experimental Results

The following test problems have been considered in this paper in order to verify the viability of the proposed approach.

1. Michalewicz's function: $f_1(x) = -\sum_{i=0}^{n} \sin(x_i) \left(\sin\left(\frac{ix_i^2}{\pi} \right) \right)^{2n}$, with $n = 10$. Decision space $D = [0, \pi]^n$.
2. Rastrigin's function: $f_2\,(x) = 10n + \sum_{i=1}^{n} \left(z_i^2 - 10\cos 2\pi z_i \right)$, with $n = 10$. Decision space $D = [-5, 5]^n$.
3. Schwefel's function: $f_3\,(x) = 418.9829n + \sum_{i=1}^{n} \left(-x_i \sin \sqrt{|x_i|} \right)$, with $n = 10$. Decision space $D = [-500, 500]^n$.
4. Sphere function: $f_4\,(x) = \sum_{i=1}^{n} z_i^2$, with $n = 10$. Decision space $D = [-100, 100]^n$.

The test problems f_5-f_8 have been obtained from f_1-f_4, by setting $n = 20$. In addition, two test problems here indicated with f_9 and f_{10} have been generated from test problems f_1 and f_3 by applying a rotation operation. More specifically, these rotated problems have been generated through the multiplication of the vector of variables by a randomly generated orthogonal rotation matrix. For each test problem, codomain range C has been estimated as the difference between the fitness value in the global optimum (when analytically known, otherwise the best fitness value detected in literature) and the average fitness value computed over 100 points pseudo-randomly generated within the decision space. Then, for all the test problems, the noisy test cases have been generated by adding to the time invariant function a zero-mean Gaussian noise characterized by a standard deviation equal to 10% of C, see [6].

For the noisy problems the real-encoded compact GAs, pe-cGA and ne-cGA respectively have been run, according to the description in [14]. Then the proposed versions including noise analysis component, i.e. pe-NAcGA and ne-NAcGA, have also been run for comparisons. All the algorithms have been run with a virtual population size $N_p = 5 \times n$. The nonpersistent elitist algorithms have been run with $\eta = 5$. For all the algorithms 50 independent runs have been performed. Each run consists of $5000 \times n$ fitness evaluations.

Experimental results in terms of average final fitness and related standard deviation are given in Table 1. The best results are highlighted in bold face. In order to estimate the actual improvements carried out by the algorithms under analysis, the following Tables refer to the "true" fitness values. By true fitness values we mean values of the fitness functions corresponding to the solution detected without the noise perturbation. Although in real world problems the true fitness values are not available, we decided to present the results in this form in order to highlight the capability of the algorithms in handling the noise conditions that we imposed.

In order to carry out a numerical comparison of the convergence speed performance, for each test problem, the average final fitness value returned by the best performing algorithm G has been considered. Subsequently, the average fitness value at the beginning of the optimization process J has also been computed. The threshold value $THR = J - 0.95(J - G)$ has then been calculated. If an algorithm succeeds during a certain run to reach the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$

Table 1. True Fitness ± standard deviation

Test Problem	pe-cGA	ne-cGA	pe-NAcGA	ne-NAcGA
f_1	-5.314e+00 ± 9.51e-01	-7.730e+00 ± 8.40e-01	-5.115e+00 ± 7.19e-01	**-7.880e+00 ± 7.71e-01**
f_2	8.048e+01 ± 1.60e+01	2.792e+01 ± 9.48e+00	6.610e+01 ± 1.02e+01	**2.424e+01 ± 9.76e+00**
f_3	1.385e+03 ± 5.13e+02	3.183e+02 ± 1.62e+02	1.662e+03 ± 3.70e+02	**1.682e+02 ± 9.00e+01**
f_4	8.189e+03 ± 3.19e+03	2.322e+03 ± 1.07e+03	4.146e+03 ± 1.66e+03	**1.560e+03 ± 7.07e+02**
f_5	-7.086e+00 ± 1.05e+00	-1.059e+01 ± 1.52e+00	-6.559e+00 ± 8.48e-01	**-1.083e+01 ± 1.25e+00**
f_6	2.130e+02 ± 2.31e+01	1.121e+02 ± 2.61e+01	1.881e+02 ± 2.14e+01	**8.501e+01 ± 2.08e+01**
f_7	4.269e+03 ± 6.57e+02	1.278e+03 ± 4.24e+02	4.738e+03 ± 6.70e+02	**1.118e+03 ± 3.88e+02**
f_8	2.806e+04 ± 4.38e+03	1.094e+04 ± 4.31e+03	2.006e+04 ± 5.20e+03	**8.164e+03 ± 3.12e+03**
f_9	-2.652e+00 ± 6.77e-01	-2.948e+00 ± 6.88e-01	-2.526e+00 ± 7.07e-01	**-2.994e+00 ± 7.45e-01**
f_{10}	7.431e+01 ± 1.50e+01	4.146e+01 ± 1.34e+01	7.168e+01 ± 1.06e+01	**3.271e+01 ± 1.02e+01**

required, for each algorithm, to reach *THR* has been computed. Subsequently, the Q-test (Q stands for Quality) described in [9] has been applied. For each test problem and each algorithm, the Q measure is computed as $Q = \frac{\bar{ne}}{R}$ where the robustness R is the percentage of successful runs. For each test problem, the smallest value equals the best performance in terms of convergence speed. The value "∞" means that $R = 0$, i.e., the algorithm never reached the *THR*. Results of the Q-test are given in Table 2. In order to prove statistical significance of the results, the Wilcoxon Rank-sum test has been applied for a confidence level of 0.95. Table 2 shows results of the test. A "+" indicates the case in which the noise analysis algorithm statistically outperforms its corresponding standard variant; a "=" indicates that the two algorithms have the same performance; a "-" indicates that the noise analysis algorithm is outperformed.

Table 2. Q-test and Wilcoxon Rank-Sum test (based on the true fitness)

Test Problem	pe-cGA	ne-cGA	pe-NAcGA	ne-NAcGA	pe	ne
f_1	8705	**1373.5778**	∞	2519.5408	=	=
f_2	∞	**1160.125**	∞	2237.5463	+	=
f_3	∞	**1310.5976**	∞	1429.0017	-	+
f_4	∞	**792.8906**	17602	1281.3081	+	+
f_5	∞	**1836.5459**	∞	3377.7511	=	=
f_6	∞	3663.6944	∞	**2865.929**	+	+
f_7	∞	**1299.6505**	∞	2677.9025	-	=
f_8	∞	**1152.051**	33071.5	1938.245	+	+
f_9	391.9136	**384.4198**	2095.2857	583.46	=	=
f_{10}	∞	**1995.9506**	∞	2416.564	=	+

For the sake of clarity an example of performance trend is shown in Fig. 2. Numerical results show that the nonpersistent schemes lead to much better results than the persistent ones. The update, on a regular basis, of the elite individual seems to be very beneficial to the algorithmic performance due to the fact that the selection of an overestimated elite can mislead the search. Since the persistent elitist scheme does not employ a replacement, the search could turn out to be jeopardized for a large portion of the evolution. In other words, the nonpersistent logic increasing the algorithmic randomization seems beneficial in the presence of noisy fitness landscapes. In order to better comprehend the role of the randomization in the presence of noise, we may considered the effect of generational vs steady-state survivor selection in EAs, see [12] and references therein, or the employment of a randomized scale factor in differential evolution, see e.g. [6]. Numerical results show also that the noise analysis component seems to have a beneficial effect on compact algorithms, especially to the nonpersistent elitist version. Table 1 shows that the proposed ne-NAcGA obtains the best results for all the test problems considered in this study and that either outperforms significantly ne-cGA or obtains similar results to ne-cGA. Due to the extra fitness evaluations (within the noise analysis), the ne-NAcGA often displays worse convergence speed performance than the ne-cGA in the scope of prematurely stopped tests like in Table 2, eventually outperforming the standard variant, see Fig. 2.

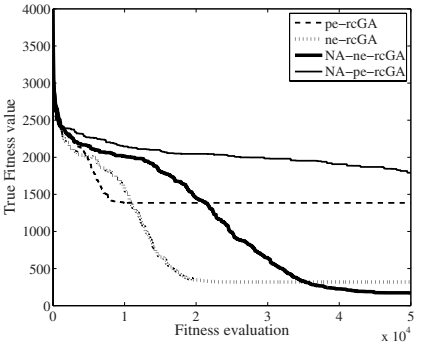


Fig. 2. Performance trend on problem f_3

4 Conclusion

This paper proposes a real-encoded compact genetic algorithm integrating a noise analysis system. The proposed algorithm can be useful for those systems engineering system characterized by a limited memory and the presence of measurement systems which affect the fitness landscape and make it noisy. The resulting algorithm integrates an adaptive system for performing the minimum amount of fitness evaluations and still reliably selecting the elite individual for the subsequent algorithmic comparison. Numerical results show that the noise analysis system efficiently enhances the performance of a standard compact genetic algorithm. The experiments have been repeated for both persistent and nonpersistent elitist schemes. Results show that the nonpersistent scheme allows a more robust behavior with respect to the persistent elitism. This fact can be interpreted in consideration that if a persistent scheme might retain as an elite an overestimated solution a persistent scheme performs a periodical update of the elite thus refurbishing the search and mitigating the effect of a possible overestimation. Future developments will consider novel possible compact structures involving other evolutionary and swarm intelligence algorithms which also employ noise analysis and compensation systems.

References

1. Ahn, C.W., Ramakrishna, R.S.: Elitism based compact genetic algorithms. IEEE Transactions on Evolutionary Computation 7(4), 367–385 (2003)
2. Arnold, D.V., Beyer, H.G.: A general noise model and its effects on evolution strategy performance. IEEE Transactions on Evolutionary Computation 10(4), 380–391 (2006)
3. Beyer, H.G., Sendhoff, B.: Functions with noise-induced multimodality: a test for evolutionary robust optimization-properties and performance analysis. IEEE Transactions on Evolutionary Computation 10(5), 507–526 (2006)

4. Branke, J., Schmidt, C.: Selection in the presence of noise. 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. 2723, pp. 766–777. Springer, Heidelberg (2003)
5. Cantú-Paz, E.: Adaptive sampling for noisy problems. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 947–958. Springer, Heidelberg (2004)
6. Caponio, A., Neri, F.: Differential evolution with noise analysis. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekart, A., Esparcia-Alcazar, A.I., Farooq, M., Fink, A., Machado, P., McCormack, J., O'Neill, M., Neri, F., Preuss, M., Rothlauf, F., Tarantino, E., Yang, S. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 715–724. Springer, Heidelberg (2009)
7. Cupertino, F., Mininno, E., Naso, D.: Elitist compact genetic algorithms for induction motor self-tuning control. In: Proceedings of the IEEE Congress on Evolutionary Computation (2006)
8. Cupertino, F., Mininno, E., Naso, D.: Compact genetic algorithms for the optimization of induction motor cascaded control. In: Proceedings of the IEEE International Conference on Electric Machines and Drives, vol. 1, pp. 82–87 (2007)
9. Feoktistov, V.: Differential Evolution. Springer, Heidelberg (2006)
10. Goh, C.K., Tan, K.C.: An investigation on noisy environments in evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation 11(3), 354–381 (2007)
11. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Transactions on Evolutionary Computation 3(4), 287–297 (1999)
12. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
13. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Dordrecht (2001)
14. Mininno, E., Cupertino, F., Naso, D.: Real-valued compact genetic algorithms for embedded microcontroller optimization. IEEE Transactions on Evolutionary Computation 12(2), 203–219 (2008)
15. Mininno, E., Neri, F.: A memetic differential evolution approach in noisy optimization. Memetic Computing (to appear, 2010)
16. Neri, F., Cascella, G.L., Salvatore, N., Kononova, A.V., Acciani, G.: Prudent-daring vs tolerant survivor selection schemes in control design of electric drives. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 805–809. Springer, Heidelberg (2006)
17. Ong, Y.S., Nair, P.B., Lum, K.Y.: Max-min surrogate-assisted evolutionary algorithm for robust design. IEEE Transactions on Evolutionary Computation 10(4), 392–404 (2006)
18. Rudolph, G.: Self-adaptive mutations may lead to premature convergence. IEEE Transactions on Evolutionary Computation 5(4), 410–414 (2001)
19. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures (2000)
20. Stagge, P.: Averaging efficiently in the presence of noise. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 188–200. Springer, Heidelberg (1998)

Author Index